全国计算机等级考试专用辅导丛书
National Computer Rank Examination

National Computer Rank Examination

全 国 计 算 机 等 级 考 试

专用辅导教程：二级C

希赛教育等考学院　陈暄 郑美芳 主编

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书由希赛教育等考学院组织编写，作为全国计算机等级考试二级 C 语言的辅导和培训指定教程。内容紧扣教育部考试中心新推出的考试大纲，通过对历年试题进行科学分析、研究、总结、提炼而成。

本书基于最新的考试大纲和历年试题，内容紧扣大纲，全面实用。全书内容涵盖了考试大纲规定的所有知识点，对考试大纲规定的内容有重点地进行了细化和深化。阅读本书，就相当于阅读了一本详细的、带有知识注释的考试大纲。准备考试的人员可通过阅读本书掌握考试大纲规定的知识，掌握考试重点和难点，熟悉内容的分布。

图书在版编目（CIP）数据

全国计算机等级考试专用辅导教程. 二级 C/陈暄，郑美芳主编.—京：电子工业出版社, 2011.1
（全国计算机等级考试专用辅导丛书）
ISBN 978-7-121-11993-4

Ⅰ. ①全… Ⅱ. ①陈… ②郑… Ⅲ. ①电子计算机－水平考试－自学参考资料
②C 语言－程序设计－水平考试－自学参考资料 Ⅳ. ①TP3

中国版本图书馆 CIP 数据核字(2010)第 199119 号

责任编辑：付　睿
印　　刷：北京中新伟业印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：860×1092　1/16　印张：17　字数：593 千字
印　　次：2011 年 1 月第 1 次印刷
印　　数：4000 册　定价：39.00 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前 言

全国计算机等级考试（NCRE）由教育部考试中心主办，面向社会，用于考查非计算机专业人员计算机应用知识与能力。考试客观、公正，得到了社会的广泛认可。

本书根据全国计算机等级考试二级 C 语言的考试大纲编写而成，本书在组织和写作上，倾注了作者们的许多精力和心血，相信能够对考生提高通过率，有效地完成“考试过关”提供帮助。考生可通过阅读本书，迅速掌握考试所涉及的知识点，进行全面梳理和系统学习考试大纲中的内容。

作者权威，阵容强大

希赛教育（www.educity.cn）专业从事人才培养、教育产品开发、教育图书出版，在职业教育方面具有极高的权威性。特别是在在线教育方面，名列前茅，希赛教育的远程教育模式得到了国家教育部门的认可和推广。

希赛教育等考学院是国内首屈一指地进行计算机等级考试在线教育的大型教育机构，在该领域取得了很好的效果。我们组织大纲制定者和阅卷组成员编写了考试辅导教材近 20 本，内容涵盖了计算机等级考试的主要级别。组织权威专家和辅导名师录制了考试培训视频教程，对历年考试进行了跟踪研究和比较研究，编写了权威的全真模拟试题。希赛教育的计算机等级考试培训采取统一教材、统一视频、统一认证教师的形式，采取线下培训与线上辅导相结合的方式，确保学员在通过考试的前提下能真正学到有用的知识。

本书由希赛教育等考学院组织编写，参加编写的人员来自大学教学一线和企业研发团队，具有丰富的教学和辅导经验，对等级考试有深入的研究，具有极强的应试技巧、理论知识、实践经验和责任心。

本书由陈暄、郑美芳主编，张友生审核了所有稿件。参加编写工作的有张友生、桂阳、何东彬、唐科萍、刘毅、胡钊源、符春、唐小娟和王勇。

在线测试，心中有数

上学吧在线测试平台（www.shangxueba.com）为考生准备了在线测试，其中有数十套全真模拟试题和考前密卷，考生可选择任何一套进行测试。测试完毕，系统自动判卷，立即给出分数。

对于考生做错的地方，系统会自动记忆，待考生第二次参加测试时，可选择“试题复习”。这样，系统就会自动把考生原来做错的试题显示出来，供考生重新测试，以加强记忆。

如此，读者可利用上学吧在线测试平台的在线测试系统检查自己的实际水平，加强考前训练，做到心中有数，考试不慌。

诸多帮助，诚挚致谢

在本书出版之际，要特别感谢教育部考试中心计算机等级考试办公室的命题专家们，编者在本书中引用了部分考试原题，使本书能够尽量方便读者的阅读。在本书的编写过程中，参考了许多相关的文献和书籍，编者在此对这些参考文献的作者表示感谢。

感谢电子工业出版社的田小康老师，他在本书的策划、选题的申报、写作大纲的确定，以及编辑、出版等方面，付出了辛勤的劳动和智慧，给予了我们很多的支持和帮助。

感谢参加希赛教育计算机等级考试辅导和培训的学员，正是他们的想法汇成了本书的源动力，他们的意见使本书更加贴近读者。

由于编者水平有限，且本书涉及的内容很广，书中难免存在错漏和不妥之处，编者诚恳地期望各位专家和读者不吝指正和帮助，对此，我们将十分感激。

互动讨论，专家答疑

希赛教育等考学院（www.csaidk.com）是中国领先的计算机等级考试在线教育网站，该网站论坛是国内人气很旺的计算机等级考试社区，在这里，读者可以和数百万考生进行在线交流，讨论有关学习和考试的问题，以及人生和职业规划的话题。希赛教育等考学院拥有强大的师资队伍，为读者提供全程的答疑服务，在线回答读者的提问。

有关本书的意见反馈和咨询，读者可在希赛教育等考学院论坛“等级考试教材”板块中的“希赛教育等考学院”栏目中与作者进行交流。

希赛教育等考学院

目 录

第1章 算法和数据结构

本章主要介绍算法、线性表、栈和队列、二叉树的概念，介绍几种常见的排序技术。结合计算机等级考试的要求，具体如表 1-1 所示。

表 1-1　考试要求

考试知识点	重要性
算法，线性表基本概念	★
栈和队列	★★★
树和二叉树	★★★★★
查找技术	★
排序技术	★★★★

1.1　算法与数据结构概述

本节的主要考点集中在算法与数据结构的基本概念上，包括算法的基本特征、复杂度，以及数据结构的表示等。

1.1.1　算法的概念

算法（Algorithm）是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。

1. 算法的基本特征

（1）有穷性。一个算法必须保证执行有限步骤之后结束。

（2）确定性。算法的每一步骤必须有确切的定义。

（3）可行性。算法原则上能够精确地运行，而且人们用笔和纸做有限次运算后即可完成。

2. 算法的基本要素

（1）算法中对数据的运算和操作：每个算法实际上是按解题要求从环境能进行的所有操作中选择合适的操作所组成的一组指令序列。

计算机可以执行的基本操作是以指令的形式描述的。一个计算机系统能执行的所有指令的集合，称为该计算机系统

的指令系统。计算机程序就是按解题要求从计算机指令系统中选择合适的指令所组成的指令序列。在一般的计算机系统中，基本的运算和操作有以下 4 类。

①算术运算：主要包括加、减、乘、除等运算。

②逻辑运算：主要包括与、或、非等运算。

③关系运算：主要包括大于、小于、等于、不等于等运算。

④数据传输：主要包括赋值、输入、输出等操作。

（2）算法的控制结构：一个算法的功能不仅仅取决于所选用的操作，而且还与各操作之间的执行顺序有关。算法中各操作之间的执行顺序称为算法的控制结构。

3. 算法设计的基本方法

计算机算法不同于人工处理的方法，下面是工程上常用的几种算法设计，在实际应用时，各种方法之间往往存在着一定的联系。

（1）递推法。递推法是利用问题本身所具有的一种递推关系求问题解的一种方法。它把问题分成若干步，找出相邻几步的关系，从而达到目的。

（2）递归法。递归指的是一个过程。函数不断引用自身，直到引用的对象已知。

（3）穷举搜索法。穷举搜索法是对可能是解的众多候选解按某种顺序进行逐一枚举和检验，并从中找出那些符合要求的候选解作为问题的解。

（4）贪婪法。贪婪法是一种不追求最优解，只希望得到较为满意解的方法。贪婪法一般可以快速得到满意的解，因为它省去了为找最优解要穷尽所有可能而必须耗费的大量时间。贪婪法常以当前情况为基础做最优选择，而不考虑各种可能的整体情况，所以贪婪法不要回溯。

（5）分治法。分治法是把一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题，直到最后子问题可以简单地直接求解，原问题的解即子问题的解的合并。

（6）动态规划法。动态规划是一种在数学和计算机科学中使用的，用于求解包含重叠子问题的最优化问题的方法。其基本思想是，将原问题分解为相似的子问题，在求解的过程中通过子问题的解求出原问题的解。动态规划的思想是多种算法的基础，被广泛应用于计算机科学和工程领域。

（7）迭代法。迭代法是数值分析中通过从一个初始估计出发寻找一系列近似解来解决问题（一般是解方程或者方程组）的过程，为实现这一过程所使用的方法统称为迭代法。

4. 良好的算法设计的要求

一个良好的算法应达到如下目标。

（1）正确性（Correctness）。算法的计算结果必须要是正确的。

（2）可读性（Readability）。可读性好有助于用户对算法的理解；不易理解的程序容易隐藏较多错误，难以调试和修改。

（3）健壮性（Robustness）。当输入数据非法时，算法也能适当地做出反应或进行处理，而不会产生莫名其妙的输出结果。

（4）效率与低存储量需求。效率指的是程序执行时，对于同一个问题如果有多个算法可以解决，执行时间短的算法效率高；存储量需求指算法执行过程中所需要的最大存储空间。

1.1.2 算法的复杂度

算法复杂度分为空间复杂度和时间复杂度。

1. 算法的时间复杂度

算法的时间复杂度，是指执行算法所需要的计算工作量。同一个算法用不同的语言实现，或者用不同的编译程序进行编译，或者在不同的计算机上运行，效率均不同。

2. 算法的空间复杂度

算法的空间复杂度是指执行这个算法所需要的内存空间。一个算法所占用的存储空间包括算法程序所占的存储空间、输入的初始数据所占的存储空间，以及算法执行中所需要的额外空间。

【例题 1】算法的空间复杂度是指（　　）。（2009 年 9 月）

A）算法在执行过程中所需要的计算机存储空间

B）算法所处理的数据量

C）算法程序中的语句或指令条数

D）算法在执行过程中所需要的临时工作单元数

例题分析

由以上定义得知，此题选 A。

1.1.3　数据结构的定义

数据结构（Data Structure）是指相互之间存在一种或多种特定关系的数据元素的集合。

数据（Data）是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。

数据元素（Data Element）是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

在一般情况下，在具有相同特征的数据元素集合中，各个数据元素之间存在有某种关系（即连续），这种关系反映了该集合中的数据元素所固有的一种结构。在数据处理领域中，通常把数据元素之间这种固有的关系简单地用前后件关系（或直接前驱与直接后继关系）来描述。

一般来说，数据元素之间的任何关系都可以用前后件关系来描述。

1. 数据的逻辑结构

数据结构是指反映数据元素之间的关系的数据元素集合的表示。通俗地说，数据结构是指带有结构的数据元素的集合。所谓结构实际上就是指数据元素之间的前后件关系。

一个数据结构应包含以下两方面信息。

（1）表示数据元素的信息。

（2）表示各数据元素之间的前后件关系。

数据的逻辑结果是对数据元素之间的逻辑关系的描述。它可以用一个数据元素的集合和在此集合中定义的若干关系来表示。用 D 表示数据元素的集合，用 R 表示数据元素之间的前后件关系，即一个数据结构可以表示为 B=(D, R)，这是一个二元关系的表示方式。

2. 数据的存储结构

数据的逻辑结构在计算机存储空间中的存放形式，称为数据的存储结构（也称为数据的物理结构）。

由于数据元素在计算机存储空间中的位置关系可能与逻辑关系不同，因此，为了表示存放在计算机存储空间中的各数据元素之间的逻辑关系（即前后件关系），在数据的存储结构中，不仅要存放各数据元素的信息，还需要存放各数据元素之间的前后件关系的信息。

一种数据的逻辑结构根据需要可以表示成多种存储结构，常用的结构有顺序、链接、索引等存储结构，而采用不同的存储结构，其数据处理的效率是不同的。因此，在进行数据处理时，选择合适的存储结构是很重要的。

1.1.4　数据结构的表示

数据结构的表示除了用二元关系表示外，还可以直观地用图形表示。

在数据结构的图形表示中，对于数据集合 D 中的每一个数据元素用中间标有元素值的方框表示，一般称之为数据结点，并简称为结点；为了进一步表示各数据元素之间的前后件关系，对于关系 R 中的每一个二元组，用一条有向线段从

前件结点指向后件结点。

在数据结构中，没有前件的结点称为根结点；没有后件的结点称为终端结点（也称为叶子结点）。

一个数据结构中的结点可能是在动态变化的。根据需要或在处理过程中，可以在一个数据结构中增加一个新结点（称为插入运算），也可以删除数据结构中的某个结点（称为删除运算）。插入与删除是对数据结构的两种基本运算。除此之外，对数据结构的运算还有查找、分类、合并、分解、复制和修改等。

1.1.5 线性结构与非线性结构

根据数据结构中各数据元素之间前后件关系的复杂程度，一般将数据结构分为两大类型：线性结构与非线性结构。

线性结构满足如下条件。

（1）有且只有一个根结点。

（2）每一个结点最多有一个前件，也最多有一个后件。

如果一个数据结构不是线性结构，则称之为非线性结构。如果在一个数据结构中一个数据元素都没有，则称该数据结构为空。线性结构与非线性结构都可以是空的数据结构。对于空的数据结构，如果对该数据结构的运算是按线性结构的规则来处理的，则属于线性结构，否则属于非线性结构。

1.2 线性表

本节主要考查线性表的基本概念，以及线性表的顺序存储方式。

1.2.1 线性表概述

线性表是一种常用的数据结构。

在实际应用中，线性表都是以栈、队列、字符串、数组等特殊线性表的形式来使用的。由于这些特殊线性表都具有各自的特性，因此，掌握这些特殊线性表的特性，对于数据运算的可靠性和提高操作效率都是至关重要的。

线性表是一个线性结构，它是一个含有 $n\geqslant 0$ 个结点的有限序列，对于其中的结点，有且仅有一个开始结点没有前驱（前件）结点但有一个后继（后件）结点，有且仅有一个终端结点没有后继结点但有一个前驱结点，其他的结点都有且仅有一个前驱结点和一个后继结点。一般地，一个线性表可以表示成一个线性序列：k_1，k_2，…，k_n，其中 k_1 是开始结点，k_n 是终端结点。

线性结构的基本特征如下。

（1）集合中必存在唯一的一个“第一元素”。

（2）集合中必存在唯一的一个“最后元素”。

（3）除最后一个元素之外，均有唯一的后继。

（4）除第一个元素之外，均有唯一的前驱。

由 n（$n\geqslant 0$）个数据元素（结点）a_1，a_2，…，a_n 组成的有限序列。数据元素的个数 n 定义为表的长度。当 n=0 时称为空表。常常将非空的线性表（n>0）记做：（$a_1,a_2,\cdots,a_n$）。

1.2.2 线性表的顺序存储

线性表的顺序存储指的是用一组地址连续的存储单元依次存储线性表的数据元素。线性表的顺序存储又叫做顺序表。

1. 线性表的顺序存储基本概念

线性表的顺序存储结构具备如下两个基本特征。

（1）线性表中的所有元素所占的存储空间是连续的。

（2）线性表中各数据元素在存储空间中是按逻辑顺序依次存放的。

假设线性表的每个元素需要占用 k 个存储单元，并且所占的存储位置 $ADR(a_i+1)$ 和第 i 个数据元素的存储位置 $ADR(a_i)$ 之间满足下列关系：

$$ADR(a_i+1)=ADR(a_i)+k$$

线性表第 i 个元素 a_i 的存储位置为：

$$ADR(a_i)=ADR(a_1)+(i-1)\times k$$

公式中 $ADR(a_1)$ 是线性表的第一个数据元素的存储位置，通常称做线性表的起始位置或基址。

在 C 语言中，通常定义一个一维数组来表示线性表的顺序存储空间，如图 1-1 所示。

a[0]	a[1]	a[2]	a[3]	a[4]

图 1-1　顺序存储

在用一维数组存放线性表时，该一维数组的长度通常要定义得比线性表的实际长度大一些，以便对线性表进行各种运算，特别是插入运算。

在线性表的顺序存储结构下，可以对线性表做以下运算。

（1）在线性表的指定位置处加入一个新的元素（即线性表的插入）。

（2）在线性表中删除指定的元素（即线性表的删除）。

（3）在线性表中查找某个（或某些）特定的元素（即线性表的查找）。

（4）对线性表中的元素进行排序（即线性表的排序）。

（5）将一个线性表分解成多个线性表（即线性表的分解）。

（6）将多个线性表合并成一个线性表（即线性表的合并）。

（7）复制一个线性表（即线性表的复制）。

（8）逆转一个线性表（即线性表的逆转）等。

2. 顺序表的基本操作

顺序表的基本操作包括插入运算和删除运算。

（1）顺序表的插入运算

线性表的插入运算是指在表的第 i（$1\leqslant i\leqslant n+1$）个位置上，插入一个新结点 x，使长度为 n 的线性表

$$(a_1,\cdots,a_{i-1},a_i,\cdots,a_n)$$

变成长度为 n+1 的线性表

$$(a_1,\cdots,a_{i-1},x,a_i,\cdots,a_n)$$

现在分析算法的复杂度。设它的值为 n。该算法的时间主要花费在循环结点后移语句上，该语句的执行次数（即移动结点的次数）是 n-i+1。由此可看出，所需移动结点的次数不仅依赖于表的长度，而且还与插入位置有关。

当 i=n+1 时，由于循环变量的终值大于初值，结点后移语句将不进行。这是最好情况，其时间复杂度为 O(1)。

当 i=1 时，结点后移语句，将循环执行 n 次，需移动表中所有结点。这是最坏情况，其时间复杂度为 O(n)。

综合以上的情况，得出算法的平均时间复杂度为 O(n)。

（2）顺序表的删除运算

线性表的删除运算是指将表的第 i（$1\leqslant i\leqslant n$）个结点删除，使长度为 n 的线性表

$$(a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$$

变成长度为 n-l 的线性表

$$(a_1,\cdots,a_{i-1},a_{i+1},\cdots,a_n)$$

该算法的时间分析与插入算法相似，结点的移动次数也是由表长 n 和位置 i 决定的。

若 i=n，则由于循环变量的初值大于终值，前移语句将不执行，无须移动结点。

若 i=1，则前移语句将循环执行 n-1 次，需移动表中除开始结点外的所有结点。这两种情况下算法的时间复杂度分别为 O(1)和 O(n)。

综合以上的情况得出在顺序表上做删除运算，平均要移动表中约一半的结点，平均时间复杂度也是 O(n)。

1.3 栈和队列

栈和队列都是特殊的线性表，其定义符合线性表的定义，其操作也类似于线性表的操作，只不过增加了一些限定而已。

1.3.1 栈的定义与操作

栈（Stack）是一种特殊的线性表。栈是只能在表的一端进行插入和删除运算的线性表，通常称插入、删除的这一端为栈顶（Top），另一端为栈底（Bottom），如图 1-2 所示。当表中没有元素时称为空栈。栈顶元素总是后插入的元素，从而也是最先被删除的元素；栈底元素总是最先被插入的元素，从而也是最后才能被删除的元素。

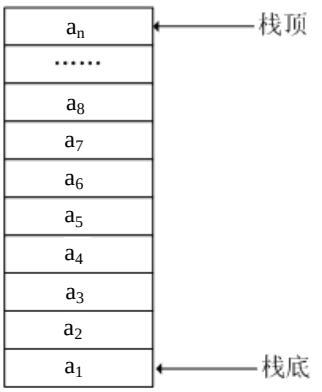


图 1-2 栈

假设栈 $S=(a_1,a_2,a_3,\cdots,a_n)$，则 a_1 为栈底元素，a_n 为栈顶元素。栈中元素按 a_1，a_2，a_3，…，a_n 的次序进栈，退栈的第一个元素应为栈顶元素。换句话说，栈的修改是按后进先出的原则进行的。因此，栈称为先进后出表（First In Last Out，FILO），或后进先出表（Last In First Out，LIFO）。

栈的操作主要有入栈运算、退栈运算（出栈）和读栈顶元素。

（1）入栈运算：入栈运算是指在栈顶位置插入一个新元素。首先将栈顶指针加 1（即 Top 加 1），然后将元素插入到栈顶指针指向的位置。当栈顶指针已经指向存储空间的最后一个位置时，说明栈空间已满，不能再进行入栈操作。这种情况称为栈“上溢”错误。

（2）退栈运算：退栈是指取出栈顶元素并赋给一个指定的变量。首先将栈顶元素（栈顶指针指向的元素）赋给一个指定的变量，然后将栈顶指针减 1（即 Top 减 1）。当栈顶指针为 0 时，说明栈空，不可进行退栈操作。这种情况称为栈的“下溢”错误。

（3）读栈顶元素：读栈顶元素是指将栈顶元素赋给一个指定的变量。这个运算不删除栈顶元素，只是将它赋给一个变量，因此栈顶指针不会改变。当栈顶指针为 0 时，说明栈空，读不到栈顶元素。

1.3.2 队列的定义与操作

队列（Queue）是只允许在一端删除，在另一端插入的顺序表，允许删除的一端叫做队头（Front），允许插入的一端叫做队尾（Rear），如图 1-3 所示。

图 1-3 队列

1. 队列的运算

当队列中没有元素时称为空队列。在空队列中依次加入元素 a_1，a_2，…，a_n 之后，a_1 是队头元素，a_n 是队尾元素。显然退出队列的次序也只能是 a_1，a_2，…，a_n，也就是说队列的修改是依先进先出的原则进行的。因此队列亦称做先进先出的线性表，或后进后出的线性表。

（1）入队操作。往队列队尾插入一个元素称为入队运算。

（2）出队操作。从队列队头删除一个元素称为出队运算。

2. 循环队列的运算

所谓循环队列，就是将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间。

在循环队列中，用队尾指针 Rear 指向队列中的队尾元素，用队头指针 Front 指向排头元素的前一个位置。因此，从排头指针 Front 指向的后一个位置直到队尾指针 Rear 指向的位置之间的所有元素均为队列中的元素。

在循环队列中进行出队、入队操作时，头尾指针仍要加 1，朝前移动。只不过当头尾指针指向向量上界（Queuesize-l）时，其加 1 操作的结果是指向向量的下界 0。

由于入队时尾指针向前追赶头指针，出队时头指针向前追赶尾指针，故队空和队满时头尾指针均相等。因此，我们无法通过 Front=Rear 来判断队列空还是满。在实际使用循环队列时，为了能区分队列满还是队列空，通常还需增加一个标志值的定义如下：当 s=0 时表示队列空，当 s=1 时表示队列非空。

（1）入队运算。入队运算是指在循环队列的队尾加入一个新元素。首先将队尾指针进一（即 Rear=Rear+1），并当 Rear=m+l 时置 Rear=1；然后将新元素插入到队尾指针指向的位置。当循环队列非空（s=l）且队尾指针等于队头指针时，说明循环队列已满，不能进行入队运算，这种情况称为“上溢”。

（2）退队运算。退队运算是指在循环队列的队头位置退出一个元素并赋给指定的变量。首先将队头指针进一（即 From=Front+1），并当 Front=m+1 时置 Front=1，然后将排头指针指向的元素赋给指定的变量。当循环队列为空（s=0）时，不能进行退队运算，这种情况称为“下溢”。

【例题 2】下列数据结果中，能够按照“先进后出”原则存取数据的是（　　）。（2009 年 9 月）

A）循环队列

B）栈

C）队列

D）二叉树

例题分析

按照“先进后出”原则存取数据的是栈。

1.4　线性链表

本节主要考查线性链表的存储方式和基本操作。

1.4.1　线性表的链式存储

在定义的链表中，若只含有一个指针域来存放下一个元素地址，称这样的链表为单链表或线性链表，如图 1-4 所示。

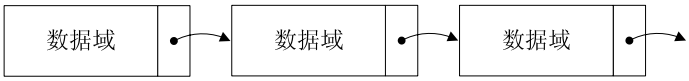


图 1-4　线性表链式存储

在链式存储方式中，要求每个结点由两部分组成：一部分用于存放数据元素值，称为数据域；另一部分用于存放指针，称为指针域。其中指针用于指向该结点的前一个或后一个结点（即前件结点或后件结点）。

1. 线性链表的存储结构

用一组任意的存储单元来依次存放线性表的结点，这组存储单元既可以是连续的，也可以是不连续的，甚至是零散分布在内存中的任意位置上的。因此，链表中结点的逻辑次序和物理次序不一定相同。为了能正确表示结点间的逻辑关系，在存储每个结点值的同时，还必须存储指示其后件结点的地址（或位置）信息，这个信息称为指针（Pointer）或链（Link）。这两部分组成了链表中的结点结构，

链表正是通过每个结点的链域将线性表的 n 个结点按其逻辑次序链接在一起。由于上述链表的每一个结点只有一个链域，故将这种链表称为单链表（Single Linked）。

显然，单链表中每个结点的存储地址存放在其前驱结点 Next 域中，而开始结点无前驱结点，故应设头指针 HEAD 指向开始结点。同时，由于终端结点无后继结点，故终端结点的指针域为空，即 NULL。

2. 线性链表的基本运算

线性链表的基本运算包括在线性链表中查找指定元素、在线性链表中插入元素、在线性链表中删除元素。

（1）在线性链表中查找指定元素

在对线性链表进行插入或删除的运算中，总是首先需要找到插入或删除的位置，这就需要对线性链表进行扫描查找，在线性链表中寻找包含指定元素的前一个结点。

在链表中，查找是否有结点值等于给定值 x 的结点，若有的话，则返回首次找到的其值为 x 的结点的存储位置；否则返回 NULL。查找过程从开始结点出发，顺着链表逐个将结点的值和给定值 x 做比较。

（2）在线性链表中插入元素

线性链表的插入是指在链式存储结构下的线性链表中插入一个新元素。

插入运算是将值为 x 的新结点插入到表的第 i-1 个结点和第 i 个结点之间。因此，我们必须首先找到 a_{i-1} 的存储位置 p，然后生成一个数据域为 x 的新结点*p，并令结点 p 的指针域指向新结点，新结点的指针域指向结点 a_i。

由线性链表的插入过程可以看出，由于插入的新结点取自于可利用栈，因此，只要可利用栈不空，在线性链表插入时总能取到存储插入元素的新结点，不会发生“上溢”的情况。而且，由于可利用栈是公用的，多个线性链表可以共享它，从而很方便地实现了存储空间的动态分配。另外，线性链表在插入过程中不发生数据元素移动的现象，只要改变有关结点的指针即可，从而提高了插入的效率。

（3）在线性链表中删除元素

线性链表的删除是指在链式存储结构下的线性链表中删除包含指定元素的结点。

删除运算是将表的第 i 个结点删去。因为在单链表中结点 a_i 的存储地址是在其直接前驱结点 a_{i-1} 的指针域 Next 中，所以我们必须首先找到 a_{i-1} 的存储位置 p，然后令 p->Next 指向 a_i 的直接后继结点，即把 a_i 从链上摘下，最后释放结点 a 的空间。

从线性链表的删除过程可以看出，从线性链表中删除一个元素后，不需要移动表中的数据元素，只要改变被删除元素所在结点的前一个结点的指针域即可。另外，可利用栈用于收集计算机中所有的空闲结点，因此，当从线性链表中删除一个元素后，该元素的存储结点就变为空闲，应将空闲结点送回到可利用栈。

1.4.2 双向链表的结构及其基本运算

双向链表也叫双链表，是链表的一种，它的每个数据结点中都有两个指针，分别指向直接后继结点和直接前驱结点，如图 1-5 所示。所以，从双向链表中的任意一个结点开始，都可以很方便地访问它的前驱结点和后继结点。

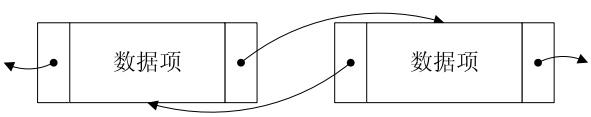


图 1-5　双向链表

1. 双向链表的基本运算

（1）插入：在双向链表的第 i 个元素前插入一个新结点的时候。第 1 步：找到待插入的位置，用指针 p 指向该结点；第 2 步：将新结点的前驱指针指向 p 结点的前一个结点；第 3 步：将 p 结点的前一个结点的后驱指针指向新的结点；第 4 步：将新结点的后驱指针指向 p 结点；第 5 步：将 p 结点的前驱指针指向新结点。

（2）删除：在双向链表中删除一个结点时，可用指针 p 指向该结点。第 1 步：将 p 结点的前一个结点的 next 指向 p 结点的下一个结点；第 2 步：将 p 的下一个结点的前驱指针指向 p 的上一个结点。

2. 双链表的结构及其基本运算

在前面所讨论的线性链表中，其插入与删除的运算虽然比较方便，但还存在一个问题，在运算过程中对于空表和对第一个结点的处理必须单独考虑，使空表与非空表的运算不统一。

因此，我们可以考虑建立这样的链表，具有单链表的特征，但又不需要增加额外的存储空间，仅对表的链接方式稍做改变，使得对表的处理更加方便灵活。从单链表可知，最后一个结点的指针域为 NULL，表示单链表已经结束。如果

将单链表最后一个结点的指针域改为存放链表中头结点（或第一个结点）的地址，就使得整个链表构成一个环，又没有增加额外的存储空间，而满足这样条件的链表叫做循环链表。

循环链表具有以下两个特点。

（1）在循环链表中增加了一个表头结点，其数据域为任意或者根据需要来设置，指针域指向线性表的第一个元素的结点。循环链表的头指针指向表头结点。

（2）循环链表中最后一个结点的指针域不是空，而是指向表头结点。即在循环链表中，所有结点的指针构成了一个环状链。

在循环链表中，只要指出表中任何一个结点的位置，就可以从它出发访问到表中其他所有的结点，而线性单链表做不到这一点。

由于在循环链表中设置了一个表头结点，因此，在任何情况下，循环链表中至少有一个结点存在，从而使空表的运算统一。

1.5　树与二叉树

本节要求考生掌握树和二叉树的基本定义，重点考查二叉树的基本性质和二叉树的遍历。

1.5.1　树的定义

树是由 n（n≥0）个结点组成的有限集合。若 n=0，称为空树；若 n＞0，则：

（1）有一个特定的称为根（Root）的结点。它只有直接后件结点，而没有直接前件结点。

（2）除根结点以外的其他结点可以划分为 m（m≥0）个互不相交的有限集合 $T_0, T_1, \cdots, T_{m-1}$，每个集合 T_i（i=0,1,⋯,m-l）又是一棵树，称为根的子树，每棵子树的根结点有且仅有一个直接前件结点，但可以有 0 个或多个直接后件结点。

图 1-6 是一棵树的示例。

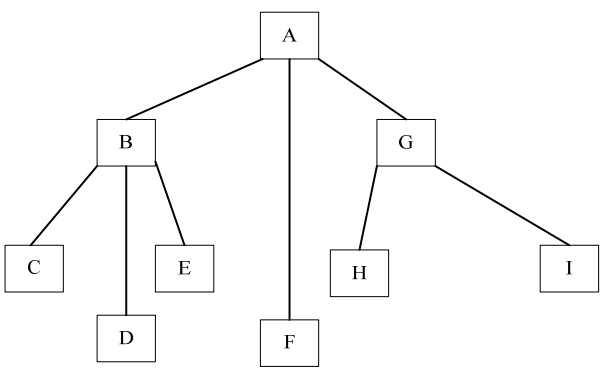


图 1-6　树的示例

1.5.2　二叉树的定义及其性质

二叉树（Binary Tree）是由 n（≥0）个结点的有限集合构成，此集合或者为空集，或者由一个根结点及两棵互不相交的左右子树组成，并且左右子树都是二叉树，如图 1-7 所示。二叉树可以是空集合，根可以有空的左子树或空的右子树。

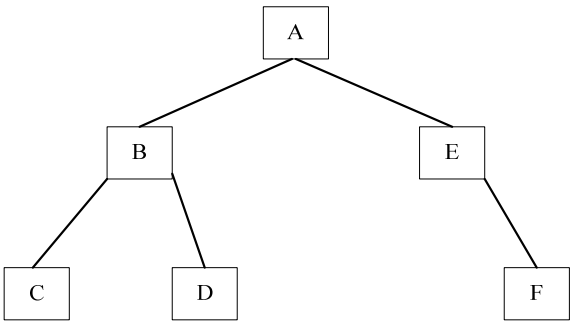


图 1-7　二叉树

1. 二叉树的特点

二叉树具有如下两个特点。

（1）非空二叉树只有一个根结点。

（2）每一个结点最多有两棵子树，且分别称为该结点的左子树和右子树。

在二叉树中，一个结点可以只有左子树而没有右子树，也可以只有右子树而没有左子树。当一个结点既没有左子树也没有右子树时，该结点即为叶子结点。

2. 满二叉树与完全二叉树

满二叉树：除最后一层外，每一层上的所有结点都有两个子结点，如图 1-8 所示。

完全二叉树：除最后一层外，每一层上的结点数均达到最大值；在最后一层上只缺少右边的若干结点，如图 1-9 所示。

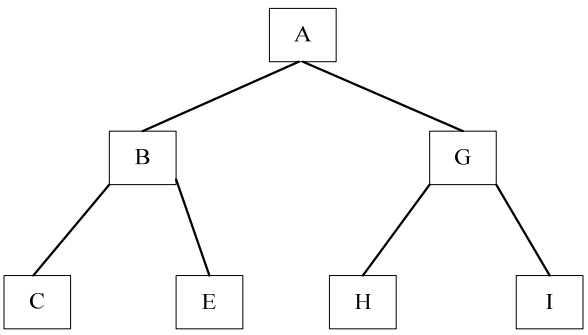


图 1-8　满二叉树

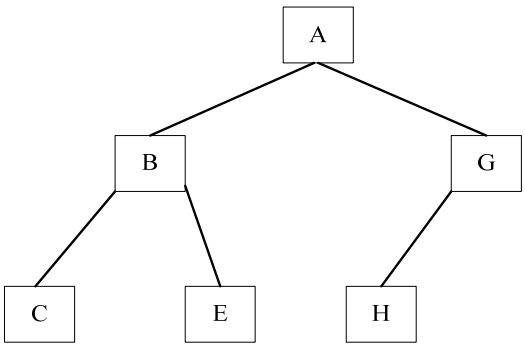


图 1-9　完全二叉树

如果有一棵具有 n 个结点的深度为 k 的二叉树，则它的每一个结点都与深度为 k 的满二叉树中编号为 1~n 的结点一一对应。

3. 二叉树的基本性质

性质 1：在二叉树的第 k 层上至多有 2^{k-1} 个结点（k≥1）。

性质 2：深度为 m 的二叉树至多有 2^m-1 个结点。

性质 3：对任何一棵二叉树，度为 0 的结点（即叶子结点）总是比度为 2 的结点多一个。

性质 4：具有 n 个结点的完全二叉树的深度至少为$[\log_2 n]+1$，其中$[\log_2 n]$表示 $\log_2 n$ 的整数部分。

性质 5：具有 n 个结点的完全二叉树深度为$[\log_2 n]+1$ 或$[\log_2(n+1)]$。

性质 6：如果对一棵有 n 个结点的完全二叉树的结点按层序编号（从第 1 层到第$[\log_2 n]+1$ 层，每层从左到右），则对任一结点 i（1≤i≤n），有：

（1）如果 i=1，则结点 i 无双亲，是二叉树的根；如果 i>1，则其双亲是结点[i/2]。

（2）如果 2i≤n，则结点 i 为叶子结点，无左孩子；否则，其左孩子是结点 2i。

（3）如果 2i+1≤n，则结点 i 无右孩子；否则，其右孩子是结点 2i+1。

【例题 3】某二叉树有 5 个度为 2 的结点及 3 个度为 1 的结点，则该二叉树中共有结点_____个。（2009 年 9 月）

例题分析

由二叉树的性质 3 得到：度数为 2 的节点个数=度数为 0 的节点个数-1，所以，度数为 0 的节点为 4 个，总的节点个数为 12 个。

4. 二叉树的存储结构

在计算机中，二叉树通常采用链式存储结构。用于存储二叉树中各元素的存储结点由两部分组成：数据域与指针域。但在二叉树中，由于每一个元素可以有两个后件结点（两个子结点），因此，用于存储二叉树的存储结点的指针域有两个：一个用于指向该结点的左子结点的存储地址，称为左指针域；另一个用于指向该结点的右子结点的存储地址，称为右指针域。

1.5.3　二叉树的遍历

所谓遍历二叉树，就是遵从某种次序，访问二叉树中的所有结点，使得每个结点仅被访问一次。

1. 前序遍历

前序遍历是指在访问根结点、遍历左子树与遍历右子树这三者中，首先访问根结点，然后遍历左子树，最后遍历右子树；并且，在遍历左右子树时，仍然先访问根结点，然后遍历左子树，最后遍历右子树。

2. 中序遍历

中序遍历是指在访问根结点、遍历左子树与遍历右子树这三者中，首先遍历左子树，然后访问根结点，最后遍历右子树。并且，在遍历左、右子树时，仍然先遍历左子树，然后访问根结点，最后遍历右子树。

3. 后序遍历

后序遍历是指在访问根结点、遍历左子树与遍历右子树这三者中，首先遍历左子树，然后遍历右子树，最后访问根结点，并且，在遍历左、右子树时，仍然先遍历左子树，然后遍历右子树，最后访问根结点。

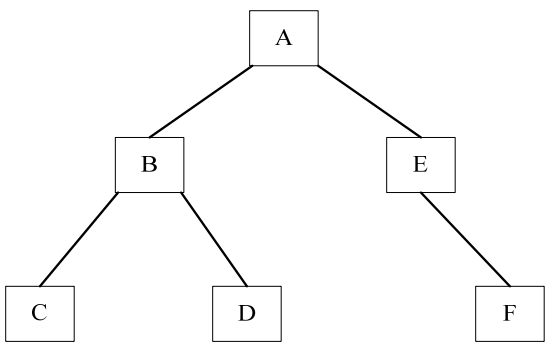


图 1-10　例 4 二叉树

【例题 4】写出图 1-10 所示的二叉树的前序、中序和后序遍历序列。

例题分析

从图 1-10 中可以得出如下结论。

前序遍历序列：ABCDEF。

中序遍历序列：CBDAEF。

后序遍历序列：CDBFEA。

【例题 5】已知二叉树的中序遍历序列为 ABCDEFG，后序遍历序列为 BDCAFGE，则前序遍历序列是什么？

例题分析

（1）步骤一。从后序中，我们可以得到根节点是 E，所以再看中序的序列：ABCDEFG，可以发现 ABCD 位于根节点的左边，而 FG 位于根节点的右边，于是得到如图 1-11 所示的图形。

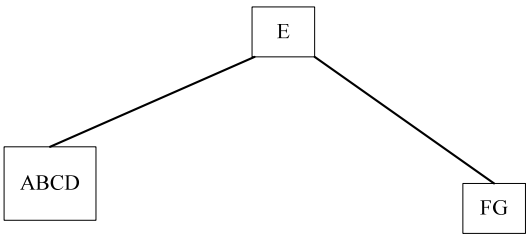


图 1-11　步骤一的图形

（2）步骤二。先来看 ABCD 这部分，然后仍然看后序序列，在后序序列中有 BDCA 这一部分，可以确定 A 是这部分的根，再看中序序列中的 ABCD，发现 BCD 都在 A 的后面。因此，可以画出图 1-12 所示的图形。

（3）步骤三。再看 BCD 这部分，从后序中看到的顺序是 BDC，所以 C 是这部分的根节点，中序序列是 BCD，可以断定 B 在 C 的左边，D 在 C 的右边。这样，得到图 1-13 所示的图形。

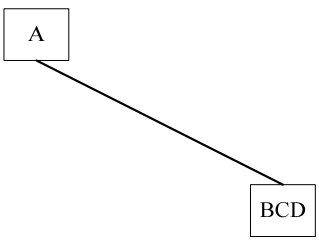


图 1-12　步骤二的图形

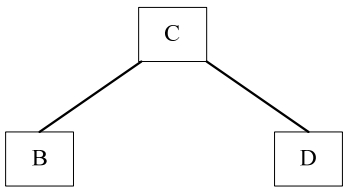


图 1-13　步骤三的图形

（4）步骤四。再看看右边的 FG 这部分，从后序序列 FG 和中序 FG 中，可以推出，G 是这部分的根节点，而 F 位于 G 的左边，如图 1-14 所示的图形。

（5）步骤五。根据以上步骤合成一个二叉树，如图 1-15 所示。

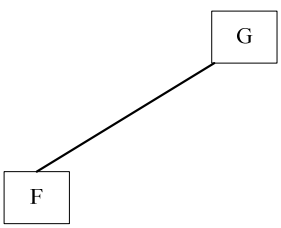


图 1-14　步骤四的图形

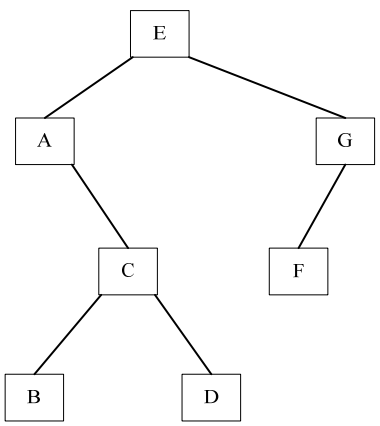


图 1-15　最后的二叉树

（6）步骤六。写出前序遍历序列：EACBDGF。

1.6　查找技术

所谓查找是指在一个给定的数据结构中查找某个指定的元素，在查找的过程中，涉及查找的方法等问题，通常，根

据不同的数据结构，应采用不同的查找方法。

1.6.1　顺序查找

顺序查找又称顺序搜索。顺序查找一般是指在线性表中查找指定的元素，其基本方法如下。

从线性表的第一个元素开始，依次将线性表中的元素与被查元素进行比较，若相等则表示找到（即查找成功）；若线性表中所有的元素与被查元素进行了比较但都不相等，则表示线性表中没有要找的元素（即查找失败）。

在进行顺序查找过程中，如果线性表中的第一个元素就是被查找元素，则只需做一次比较就查找成功，查找效率最高；但如果被查找的元素是线性表中的最后一个元素，或者被查元素根本不在线性表中，则为了查找这个元素需要与线性表中所有的元素进行比较，这是顺序查找的最坏情况。在平均情况下，利用顺序查找法在线性表中查找一个元素，大约要与线性表中一半的元素进行比较。

由此可以看出，对于大的线性表来说，顺序查找的效率是很低的。虽然顺序查找的效率不高，但在下列两种情况下也只能采用顺序查找。

（1）如果线性表为无序线性表（即表中元素的排列是无序的），则不管是顺序存储结构还是链式存储结构，都只能用顺序查找。

（2）即使有序线性表，如果线性表采用链式存储结构，也只能用顺序查找。

1.6.2　二分法查找

二分法查找只适用于顺序存储的有序表。在此所说的有序表是指线性表中的元素按值非递减排列（即从小到大，但允许相邻元素值相等）。

设有序线性表的长度为 n，被查元素为 x，则对分查找的方法如下。

将 x 与线性表的中间项进行比较。

- 若中间项的值等于 x，则说明查到，查找结束。
- 若 x 小于中间项的值，则在线性表的前半部分（即中间项以前的部分）以相同的方法进行查找。
- 若 x 大于中间项的值，则在线性表的后半部分（即中间项以后的部分）以相同的方法进行查找。

这个过程一直进行到查找成功或子表长度为 0（说明线性表中没有这个元素）为止。

1.7　排序技术

在排序技术方面，主要考查插入排序、选择排序、冒泡排序、快速排序和堆排序等方法。

1. 插入排序

每次将一个待排序的数据元素，插入到前面已经排好序的数列中的适当位置，使数列依然有序，直到待排序数据元素全部插入完为止。

【例题 6】使用插入排序对 3、4、2、1、5 按照从小到大的顺序排序。

例题分析

第一趟　3　4　2　1　5
第二趟　2　3　4　1　5
第三趟　1　2　3　4　5
第四趟　1　2　3　4　5

2. 选择排序

每一趟从待排序的数据元素中选出最小（或最大）的一个元素，顺序放在已排好序的数列的最后，直到全部待排序

的数据元素排完。

【例题 7】使用选择排序对 3、4、2、1、5 按照从小到大的顺序排序。

例题分析

第一趟　1　4　2　3　5

第二趟　1　2　4　3　5

第三趟　1　2　3　4　5

第四趟　1　2　3　4　5

3. 冒泡排序

两两比较待排序数据元素的大小，发现两个数据元素的次序相反时即进行交换，直到没有反序的数据元素为止。

设想被排序的数组 R[1..N]垂直竖立，将每个数据元素看做有重量的气泡，根据轻气泡不能在重气泡之下的原则，从下往上扫描数组 R，凡扫描到违反本原则的轻气泡，就使其向上“漂浮”，如此反复进行，直至最后任何两个气泡都是轻者在上，重者在下为止。

【例题 8】使用冒泡排序对 3、4、2、1、5 按照从小到大的顺序排序。

例题分析

第一趟　3　2　1　4　5

第二趟　2　1　3　4　5

第三趟　1　2　3　4　5

第四趟　1　2　3　4　5

4. 快速排序

在当前无序区R[1..H]中任取一个数据元素作为比较的“基准”（不妨记为X），用此基准将当前无序区划分为左右两个较小的无序区：R[1..I-1]和R[I+1..H]，且左边的无序子区中数据元素均小于等于基准元素，右边的无序子区中数据元素均大于等于基准元素，而基准X则位于最终排序的位置上，即R[1..I-1]≤X.Key≤R[I+1..H]（1≤I≤H），当R[1..I-1]和R[I+1..H]均非空时，分别对它们进行上述的划分过程，直至所有无序子区中的数据元素均已排序为止。

【例题 9】使用快速排序对 7、8、3、4、9、1 进行从小到大的排序（仅写出第一趟的结果，以第一个元素为主）

例题分析

第一趟　7　8　3　4　9　1

第一趟　1　8　3　4　9　7

第一趟　1　7　3　4　9　8

第一趟　1　7　3　4　9　8

第一趟　1　4　3　7　9　8

第一趟　1　4　3　7　9　8

一般来说，在考试的时候，只问第一趟的结果，也就是以第一个元素为主排列的结果。

5. 堆排序

堆排序是一树形选择排序，在排序过程中，将 R[1..N]看成是一棵完全二叉树的顺序存储结构，利用完全二叉树中双亲结点和孩子结点之间的内在关系来选择最小的元素。

N 个元素的序列 K_1，K_2，K_3，…，K_N 称为堆，当且仅当该序列满足特性：

$K_i \leqslant K_{2i}$，$K_i \leqslant K_{2i+1}$（$1 \leqslant i \leqslant [N/2]$）

堆实质上是满足如下性质的完全二叉树：树中任一非叶子结点的关键字均大于等于其孩子结点的关键字。

堆排序正是利用小根堆（或大根堆）来选取当前无序区中关键字最小（或最大）的记录实现排序的。我们不妨利用大根堆来排序。每一趟排序的基本操作是：将当前无序区调整为一个大根堆，选取关键字最大的堆顶记录，将它和无序区中的最后一个记录交换。这样，正好和直接选择排序相反，有序区是在原记录区的尾部形成并逐步向前扩大到整个记录区。

【例题 10】使用堆排序对 7、8、3、4、9、1 进行从小到大的排序，仅写出第一趟排序的结果。

例题分析

7　8　3　4　9　1

下标：1　2　3　4　5　6

因为有 6 个数，首先取 6/2=3，然后看看 $k_3 \leqslant k_6$ 是否成立（此处没有 k_7），因为 k_3 的值是 3，而 k_6 的值是 1，显然不满足条件，要将 k_3 和 k_6 进行交换，就变成如下形式：

7　8　1　4　9　3

下标：1　2　3　4　5　6

然后再看 $k_2 \leqslant k_4$ 和 $k_2 \leqslant k_5$ 是否成立。因为 k_2 的值是 8，k_4 的值是 4，而 k_5 的值是 9，所以，将 k_2 的值和 k_4 的值交换，得到如下序列：

7　4　1　8　9　3

下标：1　2　3　4　5　6

再看 $k_1 \leqslant k_2$ 和 $k_1 \leqslant k_3$，发现不满足，此时的 k_1 要和 k_2 与 k_3 中最小的一个进行交换，所以得到序列如下：

1　4　7　8　9　3

下标：1　2　3　4　5　6

堆建好了吗？检查每个位置是否满足上面的条件？答案是“没有”，因为此时 $k_3 \leqslant k_6$ 又不成立了，所以要进行交换，得到如下的序列：

1　4　3　8　9　7

下标：1　2　3　4　5　6

以上是第一趟排列的结果，如果要进行第二趟堆排序的话，就从剩下的 4、3、8、9、7 开始。

6. 各种排序方法的比较

各种排序方法的比较如表 1-2 所示。

表 1-2　各种排序算法的比较

方法	平均时间	最坏时间	辅助空间
冒泡，选择，插入	$O(N^2)$	$O(N^2)$	O(1)
快速排序	$O(N\log_2 N)$	$O(N^2)$	$O(N\log_2 N)$
堆排序	$O(N\log_2 N)$	$O(N\log_2 N)$	O(1)

1.8　习题

1.8.1　选择题

1. 下列叙述中错误的是（　　）。

A）一种数据的逻辑结构可以有多种存储结构　　B）数据的存储结构与数据的处理效率无关

C）数据的存储结构与数据的处理效率密切相关　　D）数据的存储结构在计算机中所占的空间不一定是连续的

2. 下列对队列的描述中正确的是（　　）。

A）队列属于非线性表　　B）队列按“先进后出”原则组织数据

C）队列在队尾删除数据　　D）队列按“先进先出”原则组织数据

3. 对长度为 N 的线性表排序，在最坏情况下，比较次数不是 N(N-1)/2 的排序方法是（　　）。

A）快速排序　　B）冒泡排序　　C）直接插入排序　　D）堆排序

4. 下列关于栈的描述中错误的是（　　）。

A）栈是先进后出的线性表　　B）栈只能顺序存储

C）栈具有记忆作用　　D）对栈进行插入、删除操作时，不需要改变栈底指针

5. 某二叉树中有 n 个度为 2 的结点，则该二叉树中的叶子结点数为（　　）。

A）n+1　　B）n-1　　C）2n　　D）n/2

6. 对图 1-16 所示的二叉树进行前序遍历的结果为（　　）。

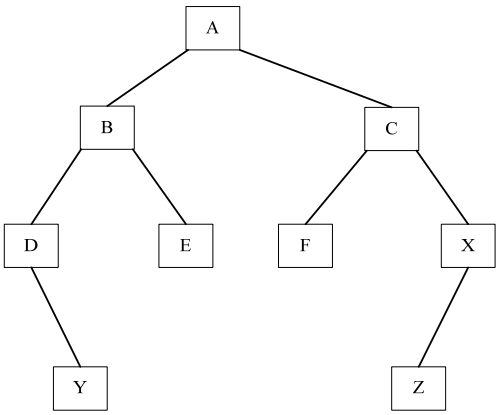


图 1-16

A）DYBEAFCZX　　B）YDEBFZXCA　　C）ABDYECFXZ　　D）ABCDEFXYZ

7. 对长度为 n 的线性表进行顺序查找，在最坏情况下需要比较的次数为（　　）。

A）125　　B）n/2　　C）n　　D）n+1

8. 下列描述中正确的是（　　）。

A）数据的逻辑结构与存储结构必定是一一对应的

B）由于计算机存储空间是向量式的存储结构，因此，数据的存储结构一定是线性结构

C）利用数组只能处理线性结构

D）以上 3 种说法都不对

9. 算法分析的目的是（　　）。

A）找出数据结构的合理性　　B）找出算法中输入和输出之间的关系

C）分析算法的易懂性和可靠性　　D）分析算法的效率以求改进

10. 按照“先进先出”原则组织数据的结构是（　　）。

A）队列　　B）栈　　C）双向链表　　D）二叉树

11. 栈和队列的共同点是（　　）。

A）都是先进先出　　B）都是先进后出

C）只允许在端点处插入和删除元素　　D）没有共同特点

12. 线性表采用链式存储时，结构的存储地址（　　）。

A）必须是不连续的　　B）连续与否均可

C）必须是连续的　　D）和头结点的存储地址相连续

13. 下列数据结构中，能用二分法进行查找的是（　　）。

A）顺序存储的有序线性表　　B）循环链表

C）二叉链表　　D）链式存储的有序线性表

14. 线性表进行二分法检索，其前提条件是（　　）。

A）线性表以顺序方式存储，并按关键码值排好序

B）线性表以顺序方式存储，并按关键码的检索频率排好序

C）线性表以链式存储，并按关键码值排好序

D）线性表以链式存储，并按关键码的检索频率排好序

15. 已知一个有序表为（13,18,34,47,50,62,83,90,115,134）。当用二分法查找值为 90 的元素时，查找成功的比较次数为（　　）。

A）1　　B）2　　C）3　　D）9

16. 线性表中经常采用的两种存储结构是（　　）。

A）顺序存储结构和链式存储结构　　B）散列方法和索引方式

C）链表存储结构和数组　　D）线性存储结构和非线性存储结构

17. 某序列的关键码序列为（33,18,25,67,82,53,95,12,70）。要按关键码值递增的顺序，采取以第一个关键码为基准元素的快速排序法，第一趟排序后关键码被放到第（　　）个位置。

A）3　　B）5　　C）7　　D）9

18. 用链表表示线性表的优点是（　　）。

A）便于随机存取　　B）花费的存储空间较顺序存储少

C）便于插入和删除操作　　D）数据元素的物理顺序与逻辑顺序相同

19. 已知数据表 A 中每个元素距其最终位置不远，为了节省时间，应采用的算法是（　　）。

A）堆排序　　B）直接插入排序

C）快速排序　　D）直接选择排序

20. 链表不具有的特点是（　　）。

A）不必事先估计存储空间　　B）可随机访问任一元素

C）插入、删除不需要移动的元素　　D）所需空间与线性表长度成正比

1.8.2　填空题

1. 在算法的 4 个特征中，算法必须能在执行有限个步骤之后终止指的是算法的_____特征。

2. 在有序列表（3,6,8,10,12,15,16,18,21,25,30）中，用二分法查找关键值 12 所需的关键码比较次数为_____。

3. 栈中允许进行插入和删除的一端叫做_____。

4. 二分法查找仅限于这样的表：表中的数据元素必须有序，其存储结构必须是_____。

第2章 程序设计结构

本章主要介绍了程序设计的方法和风格，以及结构化程序设计的内容，重点描述了面向对象程序设计的风格，结合计算机等级二级 C 语言考试要求，具体如表 2-1 所示。

表 2-1　考试要求

考试知识点	重要性
程序设计的方法与风格	★
结构化程序设计	★
面向对象的程序设计	★★

2.1　程序设计的方法与风格

程序设计风格指一个人编制程序时所表现出来的特点、习惯、逻辑思路等。

在程序设计中要使程序结构合理、清晰，形成良好的编程习惯。对程序的要求不仅是可以在机器上执行，给出正确的结果，而且要便于程序的调试和维护，这就要求编写的程序不仅自己看得懂，而且也要让别人能看懂。

随着计算机技术的发展，软件的规模扩大了，软件的复杂性也增强了。为了提高程序的可阅读性，要形成良好的编程风格。

一般来讲，程序设计风格是指编写程序时所表现出的特点、习惯和逻辑思路。程序是由人来编写的，为了测试和维护程序，往往还要阅读和跟踪程序，因此程序设计的风格总体而言应该简单和清晰，程序必须是可以理解的。

1. 数据说明的方法

（1）数据说明顺序应规范，使数据的属性更易于查找，从而有利于测试、纠错与维护。

（2）一个语句说明多个变量时，各变量名按字母排序。

（3）对于复杂的数据结构，要加注释，说明在程序实现时的特点。

2. 输入和输出

无论是批处理的输入和输出方式，还是交互式的输入和输出方式，在设计和编程时都应该考虑如下原则。

（1）输入操作步骤和输入格式尽量简单。

（2）应检查输入数据的合法性、有效性，报告必要的输入状态信息及错误信息。

（3）输入一批数据时，使用数据或文件结束标志，而不要用计数来控制。

（4）交互式输入时，提供可用的选择和边界值。

（5）当程序设计语言有严格的格式要求时，应保持输入格式的一致性。

（6）输出数据表格化、图形化。

（7）应允许默认值。

（8）当程序设计语言对输入格式有严格要求时，应保持输入格式与输入语句的一致性；给所有的输入和输出加注释，并设计输出报表格式。

2.2　结构化程序设计

常用的结构化程序设计有如下的 3 种形式。

1. 顺序结构

顺序结构是简单的程序设计，它是最基本、最常用的结构，所谓顺序执行，就是按照程序语句行的自然顺序，一条语句一条语句地执行程序，如图 2-1 所示。

2. 选择结构

选择结构又称为分支结构，它包括简单选择和多分支选择结构，这种结构可以根据设定的条件，判断应该选择哪一条分支来执行相应的语句序列，如图 2-2 所示。

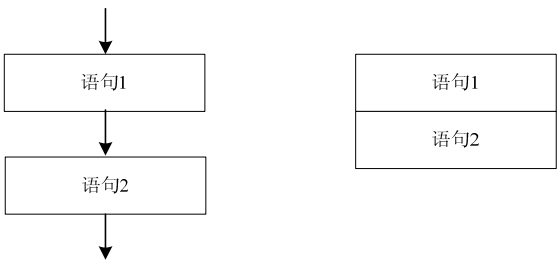


图 2-1　顺序结构程序

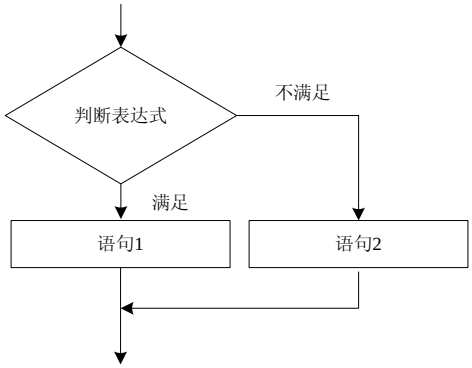


图 2-2　选择结构程序

3. 循环结构

循环结构根据给定的条件，判断是否需要重复执行某一相同的或类似的程序段，利用重复结构可简化大量的程序行。分为两类：一是先判断后执行，二是先执行后判断，如图 2-3 所示。

【例题 1】 符合结构化原则的 3 种基本控制结构是：选择结构、循环结构和_____。（2009 年 3 月）

例题分析

结构化的 3 种基本控制结构为顺序结构、选择结构和循环结构。

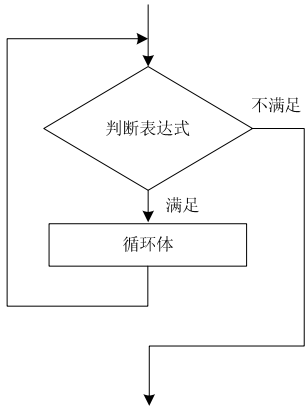


图 2-3　循环结构程序

2.3　面向对象的程序设计

对象是面向对象方法中最基本的概念。对象可以用来表示客观世界中的任何实体，也就是说，应用领域中有意义的、与所要解决的问题有关系的任

何事物都可以作为对象，它既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有明确边界意义的东西。总之，对象是对问题域中某个实体的抽象，设立某个对象就反映软件系统保存有关它的信息并具有与它进行交互的能力。

面向对象的程序设计方法中涉及的对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，它由一组表示其静态特征的属性和它可执行的一组操作组成。

对象可以做的操作表示它的动态行为，在面向对象分析和面向对象设计中，通常把对象的操作也称为方法或服务。

属性即对象所包含的信息，它在设计对象时确定，一般只能通过对象的操作来改变。

操作描述了对象执行的功能，若通过消息传递，还可以为其他对象使用。操作过程对外是封闭的，即用户只能看到这一操作实施后的结果。这相当于事先已经设计好的各种过程，只需要调用就可以了，用户不必去关心这一过程是如何编写的。事实上，这个过程已经封装在对象中，用户也看不到。这一特性即是对象的封装性。

2.3.1 面向对象特点

面向对象具有封装性、继承性和多态性。

1. 封装性

封装是一种信息隐蔽技术，它体现于类的说明，是对象的重要特性。封装使数据和加工该数据的方法（函数）封装为一个整体，以实现独立性很强的模块，使得用户只能见到对象的外特性（对象能接收哪些消息，具有哪些处理能力），而对象的内特性（保存内部状态的私有数据和实现加工能力的算法）对用户是隐蔽的。封装的目的在于把对象的设计者和对象的使用者分开，使用者不必知晓行为实现的细节，只需用设计者提供的消息来访问该对象。

2. 继承性

继承性是子类自动共享父类之间数据和方法的机制，它由类的派生功能体现。一个类直接继承其他类的全部描述，同时可修改和扩充。继承具有传递性，继承分为单继承（一个子类只有一父类）和多重继承（一个类有多个父类）。类对象是各自封闭的，如果没继承性机制，则类对象中的数据、方法就会出现大量重复。继承不仅支持系统的可重用性，而且还促进系统的可扩充性。

3. 多态性

对象根据所接收的消息而做出动作。同一消息为不同的对象接收时可产生完全不同的行动，这种现象称为多态性。利用多态性用户可发送一个通用的信息，而将所有的实现细节都留给接收消息的对象自行决定，如是，同一消息即可调用不同的方法，例如，Print 消息被发送给一图或表时调用的打印方法与将同样的 Print 消息发送给一正文文件而调用的打印方法会完全不同。多态性的实现受到继承性的支持，利用类继承的层次关系，把具有通用功能的协议存放在类层次中尽可能高的地方，而将实现这一功能的不同方法置于较低层次，这样，在这些低层次上生成的对象就能给通用消息以不同的响应。在面向对象的编程语言中可通过在派生类中重定义基类函数（定义为重载函数或虚函数）来实现多态性。

2.3.2 类和实例

类是对象的模板，即类是对一组有相同数据和相同操作的对象的定义，一个类所包含的方法和数据可描述一组对象的共同属性和行为，将属性、操作相似的对象归为类，也就是说，类是具有共同属性、共同方法的对象的集合。所以，类是对象的抽象，它描述了属于该对象类型的所有对象的性质，而一个对象则是其对应类的一个实例。

要注意的是，当使用“对象”这个术语时，既可以指一个具体的对象，也可以泛指一般的对象，但是，当使用“实例”这个术语时，必然是指一个具体的对象。

由类的定义可知，类是关于对象性质的描述，它同对象一样，包括一组数据属性和在数据上的一组合法操作。

在 UML 语言中，类的表示如下，如图 2-4 所示。

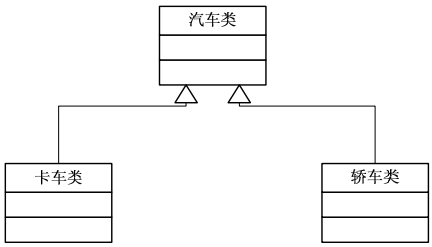


图 2-4　类图

2.3.3　消息

面向对象的世界是通过对象与对象间彼此的相互合作来推动的，对象间的这种相互合作需要一个机制协助进行，这样的机制称为“消息”。消息是一个实例与另一个实例之间传递的信息，它是对象执行某一处理或回答某一要求的信息，它统一了数据流的控制流。消息的使用类似于函数调用，消息中指定了某一个实例，一个操作名和一个参数表（可空）。接收消息的实例执行消息中指定的操作，并将形式参数与参数表中相应的值结合起来。消息传递过程中，由发送消息的对象（发送对象）的触发操作产生输出结果，作为消息传送至接收消息的对象（接收对象），引发接收消息对象的一系列的操作。所传送的消息实质上是接收对象所具有的操作/方法名称，有时还包括相应参数。

通常，一个消息由下述 3 部分组成。

（1）接收消息的对象的名称。

（2）消息标识符（也称为消息名）。

（3）零个或多个参数。

2.4　习题

2.4.1　选择题

1. 下列叙述中正确的是（　　）。

A）在面向对象的程序设计中，各个对象之间具有密切的关系

B）在面向对象的程序设计中，名个对象都是公用的

C）在面向对象的程序设计中，各个对象之间相对独立，相互依赖性小

D）上述 3 种说法都不对

2. 在面向对象方法中，实现信息隐蔽是依靠（　　）。

A）对象的继承　　B）对象的多态　　C）对象的封装　　D）对象的分类

3. 源程序的文档化不包括（　　）。

A）符号名的命名要有实际意义　　B）正确的文档形式

C）良好的视觉组织　　D）正确的程序注释

4. 在面向对象的方法中，（　　）描述的是具有相似属性与操作的一组对象。

A）属性　　B）事件　　C）方法　　D）类

5. 下列叙述中正确的是（　　）。

A）程序设计时不需要讲究风格　　B）程序中的注释是可有可无的

C）程序只要求机器读懂就可以了，不需要去关心维护的问题　　D）以上说法都不对

6. 结构化程序设计的 3 种基本结构是（　　）。

A）过程、子程序和分程序　　B）顺序、选择和重复

C）递归、堆栈和队列　　D）调用、返回和转移

7. 在面向对象的方法中，一个对象请示另一个对象为其服务的方式是通过发送（　　）。

A）调用语句　　B）命令　　C）口令　　D）消息

8. 面向对象的设计方法与传统的面向过程的设计方法有本质的不同，它的基本原理是（　　）。

A）模拟现实世界中不同事物之间的联系

B）强调模拟现实世界中的算法而不强调概念

C）使用现实世界的概念抽象地思考问题，从而自然地解决问题

D）以上说法都不对

9. 下列特征中不是面向对象方法的主要特征的是（　　）。

A）多态性　　B）继承性　　C）封装性　　D）模块化

2.4.2　填空题

1. _____的程序设计方法中涉及的对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位。

2. 在面向对象的程序设计中，从外面看只能看到对象的外部特征，而不知道也无须知道数据的具体处理方法及实现操作的算法，这称为对象的_____。

3. 在面向对象的程序设计中，类描述的是具有相似性质的一组_____。

4. _____是指编写程序时所表现出的特点、习惯和逻辑思路。

第3章 软件工程基础

本章主要介绍了软件工程的概念，结构化程序分析、设计，重点介绍了软件测试的内容。结合计算机等级考试要求，具体如表 3-1 所示。

表 3-1　考试要求

考试知识点	重要性
软件工程概念	★
结构化程序分析	★
结构化程序设计	★
软件测试	★★★
程序调试	★

3.1　软件工程基本概念

软件工程是计算机软件学科中的一个重要的内容，它对软件产业的形成和发展起着决定性的推动作用，在人类进入信息化社会时成为新信息产业的支柱。

3.1.1　软件的含义

1. 软年工程概念

计算机软件（Software）是计算机系统中与硬件相互依存的另一部分，是包括程序、数据及相关文档的完整集合。其中，程序是软件开发人员根据用户需求开发的、用程序设计语言描述的、适合计算机执行的指令（语句）序列。

软件根据应用目标的不同，是多种多样的。软件按功能可以分为：应用软件、系统软件、支撑软件（或工具软件）。应用软件是为解决特定领域的应用而开发的软件；系统软件是计算机管理自身资源，提高计算机使用效率并为计算机用户提供各种服务的软件；支撑软件是介于系统软件和应用软件之间，协助用户开发软件的工具性软件，包括辅助、支持开发和维护应用软件的工具软件。

软件的特点如下。

（1）软件是一种逻辑实体，是人类智力成果的表现形式。软件的包装、载体、印刷的文档本身都不是软件。

（2）软件的开发和制造是一个统一的过程。软件开发本身是一个过程，不是突然得到的结果；一旦软件开发完成，剩下的就只是复制分发的过程，对软件本身不再有生产制造活动。

（3）软件开发是一项经济活动，在一定的成本和时间限制下，满足用户的需求是软件开发的目标。那种仅仅为了个人兴趣进行的编程，不是工程意义上的软件开发。

（4）软件不会磨损，但可能存在错误，需要进行维护。

3.1.2 软件工程

1. 软件工程概念

软件工程是一类工程，工程是将理论和知识应用于实践的科学。就软件工程而言，它借鉴了传统工程的原则和方法，以求高效地开发高质量软件。其中应用了计算机科学、数学和管理科学。计算机科学和数学用于构造模型与算法，工程科学用于制定规范、设计范型、评估成本及确定权衡，管理科学用于计划、资源、质量和成本的管理。

软件工程这一概念，主要是针对 20 世纪 60 年代“软件危机”而提出的。自这一概念提出以来，围绕软件项目，开展了有关开发模型、方法及支持工具的研究。具体地说，在软件开发和维护过程中，软件危机主要表现在如下方面。

（1）软件需求增长得不到满足。

（2）软件生产高成本，价格昂贵。

（3）软件生产进度违法控制。

（4）软件需求定义不准确，易偏离用户需求。

（5）软件质量不易保证。

（6）软件可维护性差。

分析带来软件危机的原因，宏观方面是由于软件日益深入社会生活的各个层面，对软件需求的增长速度大大超过了技术进步所能带来的软件生产率的提高。而就每一项具体的工程任务来看，许多困难来源于软件工程所面临的任务与其他工程之间的差异，以及软件与其他工业产品的不同。

在软件开发和维护过程中，之所以存在这些严重的问题，一方面与软件本身的特点有关，例如，在软件运行前，软件开发过程的进展难衡量，质量难以评价，因此管理和控制软件开发过程相当困难；在软件运行过程中，软件维护意味着改正或修改原来的设计；另外，软件的显著特点是规模庞大，复杂度超线性增长，在开发大型软件时，要保证高质量，极端复杂困难，不仅涉及技术问题（如分析方法、设计方法、版本控制），更重要的是必须有严格而科学的管理。另一方面与软件开发和维护方法不正确有关，这是主要原因。

为了消除软件危机，通过认真研究解决软件危机的方法，认识到软件工程是使计算机软件走向工程科学的途径，逐步形成了软件工程的概念，开辟了工程学的新兴领域——软件工程学。软件工程就是试图用工程、科学和数学的原理与方法研制、维护计算机软件的有关技术及管理方法。

2. 软件生命周期

通常，将软件产品从提出、实现、使用维护到停止使用退役的过程称为软件生命周期。也就是说，软件产品从其概念开始，到该软件产品不能使用为止的整个时期都属于软件生命周期。一般可以包括可行性研究与需求分析、设计、实现、测试、交付使用及维护的活动。

还可以将软件生命周期分为软件定义、软件开发、软件运行维护 3 个阶段。

（1）可行性研究与计划制订。确定待开发的软件系统的开发目标和总体要求，给出它的功能、性能、可靠性及接口等方面的方案，制订完成开发任务的实施计划。

（2）需求分析。对待开发软件提出的需求进行分析并给出详细定义，编写软件规格说明书及初步的用户手册，提交评审。

（3）软件设计。系统设计人员和程序设计人员应该在理解软件需求的基础上，给出软件的结构、模块的划分、功能的分配及处理流程。通常，把设计阶段可分解为概要设计阶段和详细设计阶段。软件设计需要编写概要设计说明书、详细说明书和测试计划初稿，提交评审。

（4）软件实现。把软件设计转换成计算机可以接受的程序代码，完成源程序的编码，编写用户手册、操作手册等面向用户的文档，编写单元测试计划。

（5）软件测试。在设计测试用例的基础上，检验软件的各个组成部分。编写测试分析报告

（6）运行和维护。将已交付的软件投入运行，并在运行使用中不断地维护，根据新提出的需求进行必要而且可能的扩充和删改。

3.2　结构化分析方法

需求分析是整个软件开发工作最重要的一步，通过软件需求分析，才能把软件功能和性能的总体概述为具体的需求规格说明，从而形成各个软件开发阶段的基础。

需求分析所要做的工作就是深入描述软件的功能和性能，确定软件设计的限制和软件同其他系统元素的接口细节，其步骤如下。

（1）获得当前系统的物理模型。

（2）抽象出当前系统的逻辑模型。

（3）建立目标系统的逻辑模型。

（4）对所得到的模型进行进一步的补充。

3.2.1　结构化分析方法

结构化分析方法是需求分析中使用最多的一种方法，它分析的对象是数据流，它采用自顶向下、逐层分解、建立系统的处理流程，强调开发方法的结构合理性及所开发软件的结构合理性的软件开发方法。结构是指系统内各个组成要素之间的相互联系、相互作用的框架。结构化开发方法提出了一组提高软件结构合理性的准则，如分解与抽象、模块独立性和信息隐蔽等。

结构化分析方法包括数据流图（Data Flow Diagram，DFD）、数据字典（Data Dictionary，DD）、结构化语言、判定表和判定树等工具。

结构化分析的步骤如下。

（1）通过对用户的调查，以软件的需求为线索，获得当前系统的具体模型。

（2）去掉具体模型中非本质因素，抽象出当前系统的逻辑模型。

（3）根据计算机的特点分析当前系统与目标系统的差别，建立目标系统的逻辑模型。

（4）完善目标系统并补充细节，写出目标系统的软件需求规格说明。

（5）评审直到确认完全符合用户对软件的需求。

结构化分析方法的常用工具主要有数据流图、数据字典、判定树和判定表。

1. 数据流图

数据流图是描述数据处理过程的工具，是需求理解的逻辑模型的图形表示，它直接支持系统的功能建模。

数据流图从数据传递和加工的角度，来描述数据流从输入到输出的移动变换过程。数据流图中的主要图形元素与说明如图 3-1 所示。

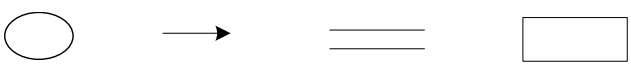


图 3-1　数据流图各要素

2. 数据字典

数据字典是结构化分析方法的核心。数据字典是对所有与系统相关的数据元素的一个有组织的列表，以及精确的、严格的定义，使得用户和系统分析员对输入、输出、存储成分和中间计算结果有共同的理解。数据字典把不同的需求文档和分析模型紧密地结合在一起，与各模型的图形表示配合，能清楚地表达数据处理的要求。

概括地说，数据字典的作用是对 DFD 中出现的被命名的图形元素的确切解释。通常数据字典包含的信息有：名称、别名、何处使用/如何使用、内容描述、补充信息等。例如，对加工的描述应包括：加工名、反映该加工层次的加工编号、加工逻辑及功能简述、输入/输出数据流等。

在数据字典的编制过程中，常使用定义式方式描述数据结构。表 3-2 给出了常用定义式符号。

表 3-2　数据字典定义式方式中出现的符号

符　号	含　义
=	表示“等于”，“定义为”，“由什么构成”
[…｜…]	表示“或”，即选择括号中用“｜”号分隔的各项中的某一项
+	表示“与”，“和”
n{}m	表示“重复”，即括号中的项要重复若干次，n，m 是重复次数的上下限
(…)	表示“可选”，即括号中的项可以没有
**	表示“注释”
..	连接符

3. 判定树

使用判定树进行描述时，应先从问题定义的文字描述中分清哪些是判定的条件，哪些是判定的结论，根据描述材料中的连接词找出判定条件之间的从属关系、并列关系、选择关系，根据它们构造判定树。

例如，在某些货物托运管理系统中，对发货情况的处理要依赖检查发货单，检查发货单受货物托运金额、欠款等条件的约束，可以使用类似分段函数的形式来描述这些约束和处理。对这些约束条件的描述，如果使用自然语言，表达容易出现不准确和不清晰。

4. 判定表

判定表与判定树相似，当数据流图中的加工要依赖于多个逻辑条件的取值，即完成该加工的一组动作是由某一组条件取值的组合而引发的，使用判定表描述比较方便。

判定表或判定树是以图形形式描述数据流图的加工逻辑，它结构简单，易读易懂。尤其遇到组合条件的判定，利用判定表或判定树可以使问题的描述清晰，而且便于直接映射到程序代码。在表达一个加工逻辑时，判定树、判定表都是好的描述工具，根据需要还可以交叉使用。如表 3-3 所示是“检查发货单”判定表。

表 3-3　“检查发货单”判定表

		1	2	3	4
条件	发货单金额	＞$500	＞$500	≤$500	≤$500
	赊欠情况	＞60 天	≤60 天	＞60 天	≤60 天
操作	不发出批准书	√			
	发出批准书		√	√	√
	发出发货单		√	√	√
	发出赊欠报告			√	

3.2.2　软件需求规格说明书

软件需求规格说明书（Software Requirement Specification，SRS）是需求分析阶段的最终成果，是软件开发中的重要文档之一。

1. 软件需求规格说明书的作用

（1）便于用户、开发人员进行理解和交流。

（2）反映出用户问题的结构，可以作为软件开发工作的基础和依据。

（3）作为确认测试和验收的依据。

2. 软件需求规格说明书的内容

软件需求规格说明书是作为需求分析的一部分而指定的可交付文档。该说明把在软件计划中确定的软件范围加以展开，制定出完整的信息描述、详细的功能说明、恰当的检验标准，以及其他与要求有关的数据。

其中，概述是从系统的角度描述软件的目标和任务。数据描述是对软件系统所必须解决的问题做出的详细说明。

功能描述是为解决用户问题所需要的每一项功能的过程细节。对每一项功能要给出处理说明和在设计时需要考虑的限制条件。性能描述是说明系统应达到的性能和应该满足的限制条件，检测的方法和标准，预期的软件响应和可能需要考虑的特殊问题。

参考文献目录描述应包括与该软件有关的全部参考文献，其中包括前期的其他文档、技术参考资料、产品目录手册及标准等。

附录部分包括一些补充资料。如列表数据、算法的详细说明、框图、图表和其他材料。

3. 软件需求规格说明书的特点

软件需求规格说明书是确保软件质量的有力措施，衡量软件需求规格说明书质量好坏的标准、标准的优先级及标准的内涵其特点如下。

（1）正确性。体现待开发系统的真实要求。

（2）无歧义性。对每一个需求只有一种解释，其陈述具有唯一性。

（3）完整性。包括全部有意义的需求，功能的、性能的、设计的、约束的，属性或外部接口等方面的需求。

（4）可验证性。描述的每一个需求都是可以验证的，即存在有限代价的有效过程验证确认。

（5）一致性。各个需求的描述不矛盾。

（6）可理解性。需求说明书必须简明易懂，尽量少包含计算机的概念和术语，以便用户和软件人员都能接受它。

（7）可修改性。SRS 的结构风格在需求有必要改变时是易于实现的。

（8）可追踪性。每一个需求的来源、流向是清晰的，当产生和改变文件编制时，可以方便地引证每一个需求。

软件需求规格说明书是一份在软件生命周期中至关重要的文件，它在开发早期就为尚未诞生的软件系统建立了一个可见的逻辑模型，它可以保证开发工作的顺利进行，因而应及时地建立并保证它的质量。

作为设计的基础和验收的依据，软件需求规格说明书应该是精确而无二义性的，需求说明书越精确，则以后出现错误、混淆、反复的可能性越小。用户能看懂需求说明书，并且发现和指出其中的错误是保证软件系统质量的关键，因而需求说明书必须简明易懂，尽量少包含计算机的概念和术语，以便用户和软件人员双方都能接受它。

3.3　结构化设计方法

在软件的需求分析阶段已经完全弄明白了软件的各种需求后，较好地解决了要让所开发的软件“做什么”的问题，并已经在软件需求说明书中详尽和充分地阐述了这些需求，下一步就是要着手解决“如何做”的问题，而软件设计就是把软件的需求分析变成软件表示的过程。

3.3.1　软件设计的基本内容

将软件的结构设计按自顶向下方式，对各个层次的过程细节和数据细节逐层细化，直到用程序设计语言的语句能够实现为止，从而最后确定整个体系结构。

模块是指把一个待开发的软件分解成若干小的简单的部分，将解决一个复杂问题时自顶向下逐层把软件系统划分成若干模块的过程。模块独立性是指，每个模块只完成系统要求的独立的子功能，并且与其他模块的联系最少且接口简单。

模块的独立程度是评价设计好坏的重要度量标准。衡量软件的模块独立性使用耦合性和内聚性两个定性的度量标准。

1. 内聚性

内聚性是一个模块内部各个元素间彼此结合的紧密程度的度量。内聚从功能角度来度量模块内的联系。

内聚有如下的种类，它们之间的内聚性由弱到强排列如下。

- 偶然内聚：指一个模块内的各处理元素之间没有任何联系。
- 逻辑内聚：指模块内执行几个逻辑上相关的功能，通过参数确定该模块完成哪一个功能。
- 时间内聚：把需要同时或顺序执行的动作组合在一起形成的模块为时间内聚模块。比如初始化模块，它顺序为变量置初值。
- 过程内聚：如果一个模块内的处理元素是相关的，而且必须以待定次序执行则称为过程内聚。
- 通信内聚：指模块内所有处理功能都通过使用公用数据而发生关系。这种内聚也具有过程内聚的特点。
- 顺序内聚：指一个模块中各个处理元素和同一个功能密切相关，而且这些处理必须顺序执行，通常前一个处理元素的输出就是下一个处理元素的输入。
- 功能内聚：指模块内所有元素共同完成一个功能，缺一不可，模块已不可再分。这是最强的内聚。

内聚性是指信息隐蔽和局部化概念的自然扩展。一个模块的内聚性越强则该模块的模块独立性越强。作为软件结构设计的设计原则，要求每一个模块的内部都具有很强的内聚性，它的各个组成部分彼此都密切相关。

2. 耦合性

耦合性是模块间互相连接的紧密程度的度量。

耦合性取决于各个模块之间接口的复杂度、调用方式及哪些信息通过接口。耦合可以分为下列几种，它们之间的耦合度由高到低排列如下。

- 内容耦合：如一个模块直接访问另一个模块的内容，则这两个模块称为内容耦合。
- 公共耦合：若一组模块都访问同一全局数据结构，则它们之间的耦合称之为公共耦合。
- 外部耦合：一组模块都访问同一全局简单变量（而不是同一全局数据结构），且不通过参数表传递该全局变量的信息，则称为外部耦合。
- 控制耦合：若一模块明显地把开关量、名字等信息送入另一模块，控制另一模块的功能，则为控制耦合。
- 标记耦合：若两个以上的模块都需要其余某一数据结构子结构时，不使用其余全局变量的方式而使用记录传递的方式，即两模块间通过数据结构交换信息，这样的耦合称为标记耦合。
- 数据耦合：若一个模块访问另一个模块，被访问模块的输入和输出都是数据项参数，即两模块间通过数据参数交换信息，则这两个模块为数据耦合。
- 非直接耦合：若两个模块没有直接关系，它们之间的联系完全是通过主模块的控制和调用来实现的，则称这两个模块为非直接耦合。非直接耦合独立性最强。

耦合性与内聚性是模块独立性的两个定性标准，耦合与内聚是相互关联的。在程序结构中，各模块的内聚性越强，则耦合性越弱。一般较优秀的软件设计，应尽量做到高内聚、低耦合，即减弱模块之间的耦合性和提高模块内的内聚性，有利于提高模块的独立性。

【例题 1】软件设计中划分模块的一个准则是（　　）。（2009 年 9 月）

A）低内聚、低耦合　　B）高内聚、低耦合　　C）低内聚、高耦合　　D）高内聚、高耦合

例题分析

本题考查软件设计中划分模块的原则，很显然“高内聚、低耦合”是划分的标准。选 B 选项。

3.3.2　结构化设计

结构化设计方法给出一组帮助设计人员在模块层次上区分设计质量的原理与技术。它通常与结构化分析方法衔接起来使用，以数据流图为基础得到软件的模块结构。结构化设计就是采用最佳的可能方法设计系统的各个组成部分及各成分之间的内部联系的技术。也就是说，结构化设计是这样一个过程，它决定用哪些方法把哪些部分联系起来，才能解决好某个具体有清楚定义的问题。

结构化设计的步骤如下。

（1）评审和细化数据流图。

（2）确定数据流图的类型。

（3）把数据流图映射到软件模块结构，设计出模块结构的上层。

（4）基于数据流图逐步分解高层模块，设计中、下层模块。

（5）对模块结构进行优化，得到更为合理的软件结构。

（6）描述模块接口。

【例题 2】下列选项中不属于结构化程序设计原则的是（　　）。（2009 年 9 月）

A）可封装　　B）自顶向下　　C）模块化　　D）逐步求精

例题分析

结构化程序设计原则包括可封装性、自顶向下分解、逐步求精的原则，所以答案选 C。

3.3.3　概要设计

概要设计的主要任务是把需求分析得到的 DFD 转换为软件结构和数据结构。设计软件结构的具体任务是：将一个复杂系统按功能进行模块划分、建立模块的层次结构及调用关系、确定模块间的接口及人机界面等。数据结构设计包括数据特征的描述、确定数据的结构特性及数据库的设计。在需求分析阶段，主要是分析信息在系统中加工和流动的情况。面向数据流的设计方法定义了一些不同的映射方法，利用这些映射方法可以把数据流图变换成结构图表示的软件结构。首先需要了解数据流图表示的数据处理的类型，然后针对不同类型分别进行分析处理。

典型的数据流类型有两种：变换型和事务型。

1. 面向变换型的数据流

变换型是指信息沿输入通路进入系统，同时由外部形式变换成内部形式，进入系统的信息通过变换中心，经加工处理以后再沿输出通路变换成外部形式离开软件系统。变换数据处理问题的工作过程大致分为 3 步，即取得数据、变换数据和输出数据，如图 3-2 所示。

2. 面向事务型的数据流

在很多软件应用中，存在某种作业数据流，它可以引发一个或多个处理，这些处理能够完成该作业要求的功能，这种数据流就叫做事务。事务型数据流的特点是接受一项事务，根据事务处理的特点和性质，选择分派一个适当的处理单元（事务处理中心），然后给出结果。这类数据流归为特殊的一类，称为事务型数据流。在一个事务型数据流中，事务中心接收数据，分析每个事务以确定它的类型，根据事务类型选取一条活动通路，如图 3-3 所示。

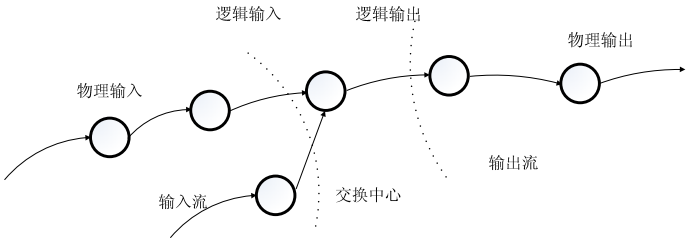


图 3-2　变换型数据流

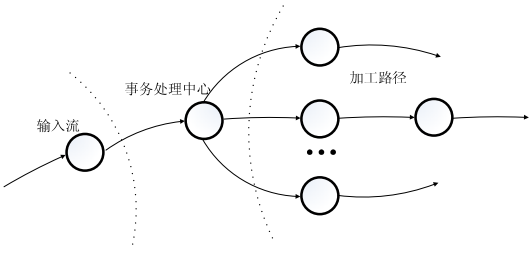


图 3-3　事务型数据流

3. 设计的过程

面向数据流设计方法的过程如下。

（1）精化 DFD。指把 DFD 转换成软件结构图前，设计人员要仔细地研究分析 DFD 并参照数据字典，认真理解其中的有关元素，检查有无遗漏或不合理之处，进行必要的修改。

（2）确定 DFD 类型。如果是变换型，确定变换中心和逻辑输入、逻辑输出的界限，映射为变换结构的顶层和第一层；如果是事务型，确定事务中心和加工路径，映射为事务结构的顶层和第一层。

（3）分解上层模块，设计中、下层模块结构。

（4）根据优化准则对软件结构求精。

（5）描述模块功能、接口及全局数据结构。

（6）复查，进入详细设计。

3.3.4 详细设计

详细设计的主要任务是设计每个模块的实现算法、所需的局部数据结构。详细设计的目标有两个：实现模块功能的算法要逻辑上正确和算法描述要简明易懂。

1. 详细设计任务

（1）为每个模块确定采用的算法，选择某种适当的工具表达算法的过程，写出模块的详细过程性描述。

（2）确定每一模块使用的数据结构。

（3）确定模块接口的细节，包括对系统外部的接口和用户界面，对系统内部其他模块的接口，以及模块输入数据、输出数据及局部数据的全部细节。

在详细设计结束时，应该把上述结果写入详细设计说明书，并且通过复审形成正式文档，是交付给下一阶段（编码阶段）的工作依据。

（4）要为每一个模块设计出一组测试用例，以便在编码阶段对模块代码（即程序）进行预定的测试，模块的测试用例是软件测试计划的重要组成部分，通常应包括输入数据、期望输出等内容。

2. 详细设计的工具

（1）图形工具。利用图形工具可以把过程的细节用图形描述出来，例如，程序流程图、N-S 图、PAD 图和 HIPO 图等。

（2）表格工具。可以用一张表来描述过程的细节，在这张表中列出了各种可能的操作和相应的条件，例如判定表等。

（3）语言工具。用某种高级语言（称之为伪码）来描述过程的细节，例如 PDL（伪码）等。

详细设计的任务，是为软件结构图中的每一个模块确定实现算法和局部数据结构，用某种特定的表达工具表示算法和数据结构的细节。表达工具可以由设计人员自由选择，但它应该具有描述过程细节的能力，而且能够使程序员在编程时便于直接翻译成程序设计语言的源程序。

3.4 软件测试

软件测试就是利用测试工具按照测试方案和流程对产品进行功能和性能测试，甚至根据需要编写不同的测试工具，设计和维护测试系统，对测试方案可能出现的问题进行分析和评估。执行测试用例后，需要跟踪故障，以确保开发的产品适合需求。

3.4.1 软件测试概述

软件测试是为了发现错误而执行程序的过程，成功的测试是为了发现至今尚未发现的错误的测试，测试的目的就是希望能以最少的人力和时间发现潜在的各种错误和缺陷，应根据开发各阶段的需求、设计等文档或程序的内部结构精心设计测试用例，并利用这些实例来运行程序，以便发现错误。

1. 软件测试的原则

（1）软件开发人员即程序员应当避免测试自己的程序。

（2）应尽早地和不断地进行软件测试。

（3）对测试用例要有正确的态度。

（4）严格执行测试计划，排除测试的随意性，以避免发生疏漏或者重复无效的工作。

（5）应当对每一个测试结果进行全面检查。一定要全面地、仔细地检查测试结果，但常常被人们忽略，导致许多错误被遗漏。

（6）妥善保存测试用例、测试计划、测试报告和最终分析报告，以备回归测试及维护之用。

2. 软件测试的分类

（1）从软件的内部结构和具体的实现角度划分，分为白盒测试、黑盒测试。

（2）从执行程序的角度划分，分为静态测试和动态测试。

（3）软件开发的过程按阶段划分，分为单元测试、集成测试、确认测试（验收测试）和系统测试。

3. 软件测试的步骤

（1）单元测试。单元测试又称模块测试，是针对软件设计的最小单位了——程序模块进行正确性检验的测试工作。其目的在于发现各模块内部可能存在的各种差错。

（2）集成测试。集成测试（联合测试）需要将所有模块按照设计要求组装成系统而进行的测试。

（3）确认测试。确认测试又称有效性测试。任务是验证软件的功能和性能及其他特性是否与用户的要求一致。

（4）系统测试。系统测试，是将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件，外设，某些支持软件、数据和人员等的其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的组装测试和确认测试。

（5）验收测试。验收测试是确保软件准备就绪，并且可以让最终用户将其用于执行软件的既定功能和任务。

3.4.2　软件测试技术

1. 静态测试

静态测试包括代码检查、静态结构分析、代码质量度量等。静态测试可以由人工进行，充分发挥人的逻辑思维优势，也可以借助软件工具自动进行。经验表明，使用人工测试能够有效地发现 30%～70%的逻辑设计和编码错误。

代码检查主要检查代码和设计的一致性，包括代码的逻辑表达的正确性，代码结构的合理性等方面。这项工作可以发现违背程序编写标准的问题，发现程序中不安全、不明确和模糊的部分，找出程序中不可移植部分、违背程序编写风格的问题，包括变量检查、命名和类型审查、程序逻辑审查、程序语法检查和程序结构检查等内容。代码检查包括代码审查、代码走查、桌面检查、静态分析等具体方式。

2. 动态测试

动态测试是基于计算机的测试，是为了发现错误而执行程序的过程。或者说，是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例（即输入数据及其预期的输出结果），并利用这些测试用例去运行程序，以发现程序错误的过程。

3. 白盒测试方法

白盒测试也称结构测试或逻辑驱动测试，它是按照程序内部结构进行测试的程序，通过测试来检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正确工作。

这一方法是把测试对象看做一个打开的盒子，测试人员依据程序内部逻辑结构的相关信息，设计或选择测试用例，对程序所有逻辑路径进行测试，通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。

白盒测试的主要方法有逻辑覆盖测试、基本路径测试等。

（1）逻辑覆盖测试

逻辑覆盖是指一系列以程序内部的逻辑结构为基础的测试用例技术。逻辑表示有判断、分支、条件等几种表示方式。

语句覆盖：选择足够的测试用例，使程序中的每一个语句至少都执行一次。

路径覆盖：执行足够的测试用例，使程序中所有的可能路径都至少执行一次。

判定覆盖：使设计的测试用例保证程序中每个判断的每个取值分支至少经历一次。

条件覆盖：设计的测试用例保证程序中每个判断的每个条件的可能取值至少执行一次。

判断—条件覆盖：设计足够的测试用例，使判断中每个条件的所有可能取值至少执行一次，同时每个判断的所有取值分支至少执行一次。

（2）基本路径测试

基本路径测试的思想是根据软件过程描述中的控制流程确定程序的环路复杂性度量，用此度量定义基本路径集合，并由此导出一组测试用例对每一条独立执行路径进行测试。

白盒测试全面了解程序内部逻辑结构、对所有逻辑路径进行测试。白盒测试是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑出发，得出测试数据，即使每条路径都测试了仍然可能有错误。第一，穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序；第二，穷举路径测试不可能查出程序中因遗漏路径而出错；第三，穷举路径测试可能发现不了一些与数据相关的错误。

4. 黑盒测试方法

黑盒测试也称功能测试，它通过测试来检测每个功能是否都能正常使用。在测试中，把程序看做一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。

黑盒测试是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。很明显，如果外部特性本身设计有问题或规格说明的规定有误，用黑盒测试方法是发现不了的。

黑盒测试方法主要有等价类划分法、边界值分析法、错误推测法、因果法等，主要用于软件确认测试。

黑盒测试法注重于测试软件的功能需求，主要试图发现下列几类错误。

（1）功能不正确或遗漏。

（2）界面错误。

（3）数据库访问错误。

（4）性能错误。

（5）初始化和终止错误等。

从理论上讲，黑盒测试只有采用穷举输入测试，把所有可能的输入都作为测试情况考虑，才能查出程序中所有的错误。实际上测试情况有无穷多个，人们不仅要测试所有合法的输入，而且还要对那些不合法但可能的输入进行测试。这样看来，完全测试是不可能的，所以我们要进行有针对性的测试，通过制定测试案例指导测试的实施，保证软件测试有组织、按步骤及有计划地进行。黑盒测试行为必须能够加以量化，才能真正保证软件质量，而测试用例就是将测试行为具体量化的方法之一。

3.5 程序的调试

所谓程序调试，是将编制的程序投入实际运行前，用手工或编译程序等方法进行测试，是修正语法错误和逻辑错误的过程。这是保证计算机信息系统正确性的必不可少的步骤。

3.5.1 步骤与方法

程序调试的步骤如下。

（1）用编辑程序把编制的源程序按照一定的书写格式送到计算机中，编辑程序会根据使用人员的意图对源程序进行增、删或修改。

（2）把送入的源程序翻译成机器语言，即用编译程序对源程序进行语法检查并将符合语法规则的源程序语句翻译成计算机能识别的“语言”。如果经编译程序检查，发现有语法错误，那就必须用编辑程序来修改源程序中的语法错误，然后再编译，直至没有语法错误为止。

（3）使用计算机中的连接程序，把翻译好的计算机语言程序连接起来，并连接成一个计算机能真正运行的程序。在

连接过程中，如果出现了连接错误，说明源程序中存在子程序的调用混乱或参数传递错误等问题。这时又要用编辑程序对源程序进行修改，再进行编译和连接，如此反复进行，直至没有连接错误为止。

（4）将修改后的程序进行试算，这时可以假设几个模拟数据去试运行，并把输出结果与手工处理的正确结果相比较，如有差异，就表明计算机的程序存在有逻辑错误。如果程序不大，可以用人工方法去模拟计算机对源程序的这几个数据进行修改处理；如果程序比较大，人工模拟显然行不通，这时只能将计算机设置成单步执行的方式，一步步跟踪程序的运行。一旦找到问题所在，仍然要用编辑程序来修改源程序，接着仍要编译、连接和执行，直至无逻辑错误为止。

程序调试主要有两种方法，即静态调试和动态调试。

3.5.2　静态调试

1. 对程序语法规则进行检查

（1）语句正确性检查。保证程序中每条语句的正确性是编写程序时的基本要求。由于程序中包含大量的语句，书写过程中由于疏忽或笔误，语句写错在所难免。对程序语句的检查应注意以下几点。

- 检查每条语句的书写是否有字符遗漏，包括必要的空格符是否都有。
- 检查形体相近的字符是否书写正确。例如，字母 o 和数字 0，书写时要有明显的分别。
- 检查函数调用时形参和实参的类型、个数是否相同。

（2）语法正确性检查。每种计算机语言都有自己的语法规则，书写程序时必须遵守一定的语法规则，否则编译时程序将给出错误信息。

（3）语句的配对检查。许多语句都是配对出现的，不能只写半个语句。另外，语句有多重括号时，每个括号也都应成对出现，不能缺左少右的。

（4）注意检查语句顺序。有些语句不仅句法本身要正确，而且语句在程序中的位置也必须正确。例如，变量定义要放在所有可执行语句之前。

2. 检查程序的逻辑结构

（1）检查程序中各变量的初值和初值的位置是否正确。我们经常遇到的是“累加”、“累乘”，其初值和位置都非常重要。用于累加的变量应取 0 作为初值或给定的初值，用于累乘的变量应赋初值 1 或给定的值。因为累加或累乘都是通过循环结构来实现的，因此这些变量赋初值语句应在循环体之外。对于多重循环结构，内循环体中的变量赋初值语句应在内循环之外；外循环体中的变量赋初值语句应在外循环之外。如果赋初值语句的位置放错了，那么将得不到预想的结果。

（2）检查程序中分支结构是否正确。程序中的分支结构都是根据给定的条件来决定执行不同的路径的，因此在设置各条路径的条件时一定要谨慎，在设置“大于”、“小于”这些条件时，一定要仔细考虑是否应该包括“等于”这个条件，更不能把条件写反。尤其要注意的是，实型数据在运算过程中会产生误差，如果用“等于”或“不等于”对实数的运算结果进行比较，则会因为误差而产生误判断，路径选择也就错了。因此在遇到判断实数 a 与 b 相等与否作为条件来选择路径时，应该把条件写成：if (fabs(a-b)<=1e-6)，而不应该写成 if (a==b)。要特别注意条件语句嵌套时，if 和 else 的配对关系。

（3）检查程序中循环结构的循环次数和循环嵌套的正确性。C 语言中可用 for 循环、while 循环、do while 循环。在给定循环条件时，不仅要考虑循环变量的初始条件，还要考虑循环变量的变化规律、循环变量变化的时间，任何一条变化都会引起循环次数的改变。

（4）检查表达式的合理与否。程序中不仅要保证表达式的正确性，而且还要保证表达式的合理性，尤其要注意表达式运算中的溢出问题。运算数值可能超出整数范围就不应该采用整型运算，否则必然导致运算结果的错误。两个相近的数不能相减，以免产生“下溢”。更要避免在一个分式的分母运算中发生“下溢”，因为编译系统常把下溢做零处理，因此分母中出现下溢时会产生“被零除”的错误。由于表达式不合理而引起的程序运行错误往往很难查找，会增加程序调试的难度。因此，认真检查表达式的合理性，是减少程序运行错误，提高程序动态调试效率的重要方面。

（5）程序的静态调试是程序调试非常重要的一步。初学者应培养自己静态检查的良好习惯，在上机前认真做好程序的静态检查工作，从而节省上机时间，使有限的机时充分发挥作用。

3.5.3 动态调试

在静态调试中可以发现和改正很多错误，但由于静态调试的特点，有一些比较隐蔽的错误还不能检查出来。只有上机进行动态调试，才能够找到这些错误并改正它们。

1. 编译过程中的调试

编译过程除了将源程序翻译成目标程序外，还要对源程序进行语法检查。如果发现源程序有语法错误，系统将显示错误信息。用户可以根据这些提示信息查找出错误性质，并在程序中出错之处进行相应的修改。有时我们会发现编译时有几行错误信息都是一样的，检查这些行本身没有发现错误，这时要仔细检查与这些行有关的名字、表达式是否有问题。例如，因为程序中数组说明语句有错，这时，那些与该数组有关的程序行都会被编译系统检查出错。在这种情况下，用户只要仔细分析一下，修改了数组说明语句的错误，许多错误就会同时没有了。对于编译阶段的调试，要充分利用屏幕给出的错误信息，对它们进行仔细分析判断。只要注意总结经验，使程序通过编译是不难做到的。

2. 连接过程中的调试

编译通过后要进行连接。连接过程也有查错功能，它将指出外部调用、函数之间的联系及存储区设置等方面的错误。如果连接时有这类错误，编译系统也会给出错误信息，用户要对这些信息仔细判断，从而找出程序中的问题并改正。连接时较常见的错误有以下几类。

（1）某个外部调用有错，通常系统明确提示了外部调用的名字，只要仔细检查各模块中与该名字有关的语句，就不难发现错误。

（2）找不到某个库函数或某个库文件，这类错误是由于库函数名写错、疏忽了某个库文件的连接等。

（3）某些模块的参数超过系统的限制。如，模块的大小、库文件的个数超出要求等。

引起连接错误的原因很多，而且很隐蔽，给出的错误信息也不如编译时给出的直接、具体。因此，连接时的错误要比编译时的错误更难查找，需要仔细分析判断，而且对系统的限制和要求要有所了解。

3. 运行过程中的调试

运行过程中的调试是动态调试的最后一个阶段。这一阶段的错误大体可分为两类。

（1）运行程序时给出出错信息。运行时出错多与数据的输入、输出格式有关，与文件的操作有关。如果给出的数据格式有错，这时要对有关的输入输出数据格式进行检查，一般容易发现错误。如果程序中的输入输出函数较多，则可以在中间插入调试语句，采取分段隔离的方法，很快就可以确定错误的位置了。如果是文件操作有误，也可以针对程序中的有关文件的操作采取类似的方法进行检查。

（2）运行结果不正常或不正确。运行结果不正确，这样的调试结果一般来说是程序设计方面出现了一定的问题。

3.6 习题

3.6.1 选择题

1. 下列工具中为需求分析常用工具的是（　　）。

A）PAD　　B）BFD　　C）N-S　　D）DFD

2. 软件设计一般分为两步完成，它们是（　　）。

A）概要设计与详细设计　　B）数据设计与接口设计

C）软件结构设计与数据设计　　D）过程设计与数据设计

3. 软件生命周期中所花费最多的阶段是（　　）。

A）详细设计　　B）软件编码　　C）软件测试　　D）软件维护

4. 在软件开发中，需求分析阶段产生的主要文档是（　　）。

A）可行性分析报告　　B）软件需求规格说明书　　C）概要设计说明书　　D）集成测试计划

5. 下列关于软件工程的描述中正确的是（　　）。

A）软件工程只是解决软件项目的管理问题

B）软件工程只是解决产品的生产率问题

C）软件工程的主要思想是强调在软件开发过程中需要应用工程化原则

D）软件工程只是解决软件开发中的技术问题

6. 在软件设计中，不属于过程设计工具的是（　　）。

A）PDL　　B）PAD 图　　C）N-S 图　　D）DFD 图

7. 下列描述中正确的是（　　）。

A）软件测试应该由程序开发者来完成　　B）程序经调试后一般不需要测试

C）软件维护只包括对程序代码的维护　　D）以上 3 种说法都不对

8. 下面不属于软件工程要素的是（　　）。

A）工具　　B）过程　　C）方法　　D）环境

9. 软件工程学一般包括软件开发技术和软件工程管理两方面的内容，软件工程经济学是软件工程管理的技术内容之一，它专门研究（　　）。

A）软件开发的方法学　　B）软件开发技术和工具　C）软件成本效益分析　　D）计划、进度和预算

10. 软件需要分析阶段的工作，可以分为 4 个方面，即需求获取、需求分析、编写需求规格说明书及（　　）。

A）阶段性能报告　　B）需求评审　　C）总结　　D）都不正确

11. 设计软件结构是软件生命周期的（　　）。

A）软件定义期　　B）软件开发期　　C）软件维护期　　D）以上 3 个都不是

12. 数据流图用于抽象描述一个软件的逻辑模型，数据流图由一些特定的图形符号组成。下列不符合数据流图的是（　　）。

A）控制流　　B）加工　　C）数据存储　　D）源程序

13. 下列叙述中正确的是（　　）。

A）程序设计就是编制程序　　B）程序测试必须由程序员自己去完成

C）程序经调试改错后还应进行再测试　　D）程序经调试改错后不必进行再测试

14. 为了提高软件的独立性，模块之间最好是（　　）。

A）控制耦合　　B）公共耦合　　C）内容耦合　　D）数据耦合

15. 需要分析阶段的任务是（　　）。

A）软件开发方法　　B）软件开发工具书　　C）软件开发费用　　D）软件系统功能

16. 下列选项中不属于软件生命周期开发阶段任务的是（　　）。

A）软件测试　　B）概要设计成本　　C）软件维护　　D）详细设计

3.6.2　填空题

1. 软件产品从提出、实现、使用维护到停止使用退役的过程称为_____。

2. 等价类划分法是_____测试的常用方法。

3. 软件生命周期包括 8 个阶段。为使各时期的任务明确，可以分为以下 3 个时期：软件定义期、软件开发期、软件维护期。编码和测试属于_____期。

4. 软件是程序、数据和_____的集合。

5. 全面支持软件开发过程的软件工具集合称为_____。

6. 软件设计模块的目的是_____程序设计的复杂性。

第4章 数据库设计基础

本章主要介绍了数据库的基本概念、数据模型、关系代数等内容。结合计算机等级考试二级 C 语言要求，具体如表 4-1 所示。

表 4-1　考试要求

考试知识点	重要性
数据库的基本概念	★
数据模型	★★★
关系代数	★★
数据库的设计与管理	★★

4.1　数据库的基本概念

在计算机系统开发过程中，数据库是非常重要的组成部分，数据库就是存放数据的仓库，人们在收集并抽取了大量数据之后，应将其保存起来，以供进一步的发展，而这些数据保存的地方就是数据库。

4.1.1　数据和信息

数据处理是计算机应用的主流，特别是关系数据库管理系统 DBSEIII推出之后，数据库技术就成为计算机应用者必须掌握的技能。数据和信息是数据处理中两个最基本的概念。

数据是客观事物的属性值，反映客观事物的特征；信息则是由客观事物得到的、使人们能够认知客观事物的各种消息、情报、数据、信号所包括的内容。

数据是物理的，信息是观念性的。信息可以数据化，数据代表信息，但二者是有区别的。信息是现实世界中的事物反映到人们头脑并经过识别、选择、命名和分类等综合分析形成印象和概念并产生的认识。数据是现实世界中的客观现象经过信息抽象后的表示形式。

数据不仅包括数值数据，而且包括各种非数值数据。

对计算机而言，数据则是指能够用计算机处理的数字、字母和符号，包括数值数据和非数值数据。

4.1.2　数据处理、数据库与数据库管理系统

数据处理包括对数据进行收集、存储、传送、整理、检索、计算、输出等各种加工和管理。

数据库（Data Base，DB）是数据的集合，它有统一的结构形式存放于统一的存储介质内，是多种应用数据的集成，并可被各种应用程序所共享。

数据库中的数据具有集成、共享的特点，数据库亦集中了各种应用的数据，进行统一的构造与存储，而使它们可被不同应用程序所使用。

1. 数据库系统

数据库系统（Database System，DBS）由如下几部分组成：数据库、数据库管理系统、数据库管理员、硬件平台、软件平台。这 5 个部分构成了一个以数据库为核心的完整的运行实体，称为数据库系统。

在数据库系统中，硬件平台包括如下。

- 计算机：它是系统中硬件的基础平台，目前常用的有微型机、小型机、中型机、大型机。
- 网络：以建立在网络上为主，而其结构形式又以客户、服务器（C/S）方式与浏览器/服务器（B/S）方式为主。

在数据库系统中，软件平台包括如下。

- 操作系统：它是系统的基础软件平台，目前常用的有各种 UNIX（包括 Linux）与 Windows 两种。
- 数据库系统开发工具：为开发数据库应用程序所提供的工具，包括可视化开发工具 VB.net、Eclipse、SQL 2005 等。

2. 数据库管理系统

数据库管理系统（Database Management System，DBMS）是数据库的机构，用来帮助用户建立、使用、管理、加工和维护数据库而配置的专用系统软件，它在操作系统的支持下对数据库进行统一的管理和控制，并且其结构复杂，因此需要提供管理工具。数据库管理系统是数据库系统的核心，它主要有如下几方面的具体功能。

（1）数据模式定义。数据库管理系统负责为数据库构建模式，也就是为数据库构建起数据框架。

（2）数据存取的物理构建。数据库管理系统负责为数据模式的物理存取及构建提供有效的存取方法与手段。

（3）数据操纵。数据库管理系统为用户使用数据库中的数据提供方便，它一般提供查询、插入、修改及删除数据的功能。此外，它自身还具有做简单算术运算及统计的能力，而且还可以与某些过程性语言相结合，使其具有强大的过程性操作能力。

（4）数据的完整性、安全性定义与检查。数据库中的数据具有内在语义上的关联性与一致性，它们构成了数据的完整性，数据的完整性是保证数据库中数据正确的必要条件，因此必须经常检查以维护数据的正确。

（5）数据库的并发控制与故障恢复。数据库是一个集成、共享的数据集合体，它能为多个应用程序服务，所以就存在多个应用程序对数据库的并发操作。在并发操作中如果不加控制和管理，多个应用程序间就会相互干扰，从而对数据库中的数据造成破坏。因此，数据库管理系统必须对多个应用程序的并发操作做必要的控制以保证数据不受破坏，这就是数据库的并发控制。

（6）数据的服务。数据库管理系统提供对数据库中的数据的多种服务功能，如数据复制、转存、重组、性能监测、分析等。

为完成以上 6 个功能，数据库管理系统一般提供相应的数据语言（Data Language），它们分别如下。

- 数据定义语言（Data Definition Language，DDL）。该语言负责数据的模式定义与数据的物理存取构建。
- 数据操纵语言（Data Manipulation Language，DML）。该语言负责数据的操纵，包括查询及增、删、改等操作。
- 数据控制语言（Date Control Language， DCL）。该语言负责数据完整性、安全性的定义与检查，以及并发控制、故障恢复等功能，包括系统初启程序、文件读写与维护程序、存取路径管理程序、缓冲区管理程序、安全性控制程序、完整性检查程序、并发控制程序、事务管理程序、运行日志管理程序、数据库恢复程序等。

3. 数据库应用系统

数据库应用系统（Database Application System，DBAS）是程序员根据用户的需要在数据库管理系统的支持下，用数据库管理系统提供的命令编写、开发并能够在数据库管理系统的支持下运行的程序和数据库的总称。如各种财务管理系统、人事管理系统、物质管理系统等。它包含 5 个主要方面，分别是：硬件、操作系统、数据库管理系统、应用开发工具、应用系统。

4.1.3 数据库系统的发展

数据管理发展至今已经经历了 3 个阶段：人工管理阶段、文件系统阶段和数据库系统阶段。人工管理阶段在 20 世纪 50 年代中期以前，主要用于科学计算，硬件无磁盘，直接存取，软件没有操作系统；20 世纪 50 年代后期到 20 世纪 60 年代中期，进入文件系统阶段；20 世纪 60 年代之后，数据管理进入数据库系统阶段。随着计算机应用领域不断扩展，数据库系统的功能和应用范围也愈来愈广，到目前已成为计算机系统的基本及主要的支撑软件。

1. 文件系统阶段

文件系统是数据库系统发展的初级阶段，它提供了简单的数据共享与数据管理能力，但是它无法提供完整的、统一的、管理和数据共享的能力。由于它的功能简单，因此它附属于操作系统而不成为独立的软件，目前一般将其看成仅是数据库系统的雏形，而不是真正的数据库系统。

2. 层次数据库与网状数据库系统阶段

从 20 世纪 60 年代末期起，真正的数据库系统——层次数据库与网状数据库开始发展，它们为统一管理与共享数据提供了有力支撑，这个时期数据库系统蓬勃发展形成了有名的“数据库时代”。但是这两种关系也存在不足，主要是它们脱胎于文件系统，受文件的物理影响较大，对数据库使用带来诸多不便，同时，此类系统的数据模式构造烦琐，不宜于推广使用。

3. 关系数据库系统阶段

关系数据库系统出现于 20 世纪 70 年代，在 20 世纪 80 年代得到蓬勃发展，并逐渐取代前两种系统。关系数据库系统结构简单、使用方便、逻辑性强、物理性少，因此在 20 世纪 80 年代以后一直占据数据库领域的主导地位，但是由于此系统来源于商业应用，适合于事务处理领域而对非事务领域应用受到限制，因此在 20 世纪 80 年代末期兴起与应用技术相结合的各种专用数据库系统。

目前，数据库技术也与其他信息技术一样在迅速发展之中，计算机处理能力的增强和越来越广泛的应用是促进数据库技术发展的重要动力。一般认为，未来的数据库系统应支持数据管理、对象管理和知识管理，应该具有面向对象的基本特征。在关于数据库的诸多新技术中，下面 3 种是比较重要的。

- 面向对象数据库系统：用面向对象的方法构筑面向对象的数据模型，使其具有比关系数据库系统更为通用的能力。
- 知识库系统：用人工智能中的方法，特别是用谓词知识表示方法构筑数据模型，使其模型具有特别通用的能力。
- 关系数据库系统的扩充：利用关系数据库系统做进一步扩展，使其在模型的表达能力与功能上有进一步的加强，如与网络技术相结合的 Web 数据库、数据仓库及嵌入式数据库等。

数据库技术是在文件系统基础上发展产生的，二者都以数据文件的形式组织数据，但由于数据库系统在文件系统之上加入了 DBMS 对数据进行管理，从而使得数据库系统具有以下特点。

- 数据的集成性。
- 数据的高共享性与低冗余性。
- 数据独立性。
- 数据的统一管理与控制。

4.1.4 数据库系统的内部结构体系

数据库系统的内部结构主要体现在三级模式和两级映射上。

1. 数据库系统的三级模式

数据模式是数据库系统中数据结构的一种表示形式，它具有不同的层次与结构方式，如图 4-1 所示。

（1）概念模式。概念模式（Conceptual Schema）是数据库系统中全局数据逻辑结构的描述，是全体用户（应用）公共数据视图。此种描述是一种抽象的描述，它不涉及具体的硬件环境与平台，也与具体的软件环境无关。

概念模式主要描述数据的概念记录类型及它们间的关系，它还包括一些数据间的语义约束，对它的描述可用 DBMS 中的 DDL 语言定义。

（2）外模式。外模式（External Schema）也称子模式（Subschema）或用户模式（User’s schema）。它是用户的数据

视图，也就是用户所见到的数据模式，它由概念模式推导而出。概念模式给出了系统全局的数据描述而外模式则给出每个用户的局部数据描述。一个概念模式可以有若干个外模式，每个用户只关心与它有关的模式，这样不仅可以屏蔽大量无关信息，而且有利于数据保护。在一般的 DBMS 中都提供有相关的外模式描述语言（外模式 DDL）。

（3）内模式。内模式（Internal Schema）又称物理模式（Physical Schema），它给出了数据库物理存储结构与物理存取方法，如数据存储的文件结构、索引、集簇及 Hash 等存取方式与存取路径，内模式的物理性主要体现在操作系统及文件级上，它还未深入到设备级上（如磁盘及磁盘操作）。内模式对一般用户是透明的，但它的设计直接影响数据库的性能。DBMS 一般提供相关的内模式描述语言（内模式 DDL）。

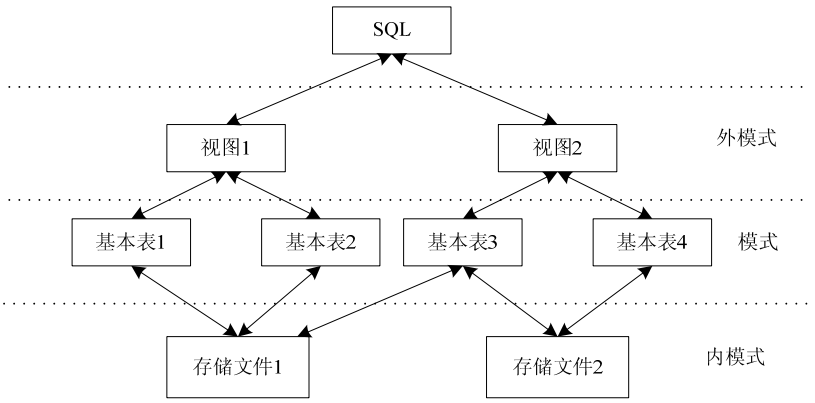


图 4-1　三级模式图

模式的 3 个级别层次反映了模式的 3 个不同环境及它们的不同要求，其中内模式处于最底层，它反映数据在计算机物理结构中的实际存储形式，概念模式处于中层，它反映了设计者的数据全局逻辑要求，而外模式处于最外层，它反映了用户对数据的要求。

2. 数据库系统的两级映射

数据库系统的三级模式是对数据的 3 个级别抽象，它把数据的具体物理实现留给物理模式，使用户与全局设计者不必关心数据库的集体实现与物理背景；同时，它通过两级映射建立了模式间的联系与转换，使得概念模式与外模式虽然并不具备物理存在，但是也能通过映射获得其实体。此外，两级映射也保证了数据库系统中数据的独立性，亦即数据的物理组织改变与逻辑概念级改变相互独立，使得只要调整映射方式而不必改变用户模式，如图 4-2 所示。

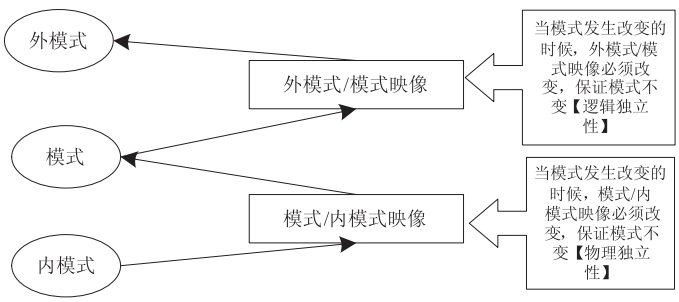


图 4-2　三级模式映射

（1）概念模式到内模式的映射。该映射给出了概念模式中的全局逻辑结构到数据的物理存储结构间的对应关系，此种映射一般由 DBMS 实现。

（2）外模式到概念模式的映射。概念模式是一个全局模式，而外模式是用户的局部模式。一个概念模式中可以定义多个模式，而每个外模式是概念模式的一个基本视图。外模式到概念模式的映射给出了外模式与概念模式的对应关系，这种映射一般也是由 DBMS 来实现的。

4.2　数据模型

数据库中的数据模型可以将复杂的现实世界要求反映到计算机数据库中的物理世界，从现实世界来看，用户为了某

种需要，需将现实世界中的部分需求用数据库实现，而数据模型（Data Model）是现实世界中数据特征的抽象，为数据库系统的信息表达与操作提供一个抽象的框架。

4.2.1 数据模型概述

数据模型所描述的内容包括 3 个部分：数据结构、数据操作和数据约束。

（1）数据结构：数据模型中的数据结构主要描述数据的类型、内容、性质及数据间的联系等。数据结构是数据模型的基础，数据操作和约束都建立在数据结构上。不同的数据结构具有不同的操作和约束。

（2）数据操作：数据模型中的数据操作主要描述在相应的数据结构上的操作类型和操作方式。

（3）数据约束：数据模型中的数据约束主要描述数据结构内数据间的语法、词义联系，它们之间的制约和依存关系，以及数据动态变化的规则，以保证数据的正确、有效和相容。

数据模型的分类如下。

（1）概念数据模型简称概念模型，它是一种面向客观世界、面向用户的模型，它与具体的数据库管理系统无关，与具体的计算机平台无关。概念模型着重于对客观世界复杂事物的结构描述及它们之间的内在联系的刻画。概念模型是整个数据模型的基础。

（2）逻辑数据模型又称数据模型，它是一种面向数据库系统的模型，该模型着重于数据库系统一级的实现。概念模型只有在转换成数据模型后才能在数据库中得以表示。目前，逻辑数据模型也有很多种，较为成熟并先后被人们大量使用过的有：层次模型、网状模型、关系模型、面向对象模型等。

（3）物理数据模型又称物理模型，它是一种面向计算机物理表示的模型，此模型给出了数据模型在计算机上物理结构的表示。

目前，数据库领域中最常用的数据模型有 4 种。

（1）层次模型（Hierarchical Model）：用层次结构表示实体类型及实体间联系的数据模型。

（2）网状模型（Network Model）：用有向图结构表示实体类型及实体间联系的数据结构模型。

（3）关系模型（Relation Model）：关系数据模型是以关系数学理论为基础的，用二维表结构来表示实体及实体之间联系的模型。

（4）面向对象模型（Object Oriented Model）：对象模型表示了静态的、结构化的系统数据性质，描述了系统的静态结构，它是从客观世界实体的对象关系角度来描述的，表现了对象的相互关系。

4.2.2 E-R 模型

概念模型中被广泛使用的模型是 E-R 模型（Entity-Relationship Model），该模型将现实世界的要求转化成实体、联系、属性等几个基本概念，以及它们间的两种基本连接关系，并且可以用一种图非常直观地表示出来。它是数据库系统开发中最重要的模型。

1. E-R 模型的基本概念

（1）实体。现实世界中的事物可以抽象成为实体，实体是概念世界中的基本单位，它们是客观存在的且又能相互区别的事物。凡是有共性的实体可组成一个集合称为实体集（Entity Set）。如小赵、小李是实体，他们又均是学生，因而组成一个实体集。

（2）属性。现实世界中的事物均有一些特性，这些特性可以用属性来表示，属性刻画了实体的特征。一个实体往往可以有若干个属性。每个属性可以有值，一个属性的取值范围称为该属性的值域（Value Domain）或值集（Value Set）。如小赵年龄取值为 17，小李为 19。

（3）联系。现实世界中事物间的关联称为联系。在概念世界中联系反映了实体集间的一定关系。

两个实体集间的联系实际上是实体集间的函数关系，这种函数关系可以有下面几种。

（1）一对一（One to One）的联系，简记为 1∶1。这种函数关系是常见的函数关系之一，如学校与校长间的联系，一个学校与一个校长间相互一一对应，如图 4-3 所示。

（2）一对多（One to Many）或多对一（Many to One）联系，简记为 1∶M（1∶m）或 M∶1(m∶1)。这两种函数关系实际上是一种函数关系，如班主任与其学生的联系是多对一的联系（反之，则为一对多的联系），即多个学生对应一个班主任，如图 4-4 所示。

图 4-3　一对一关系图

图 4-4　一对多关系图

（3）多对多（Many to Many）联系，简记为 M∶N 或 m∶n。这是一种较为复杂的函数关系，如课程与学生这两个实体集间的联系是多对多的，因为一门课程可以有多个学生学习，而一个学生又可以学习多门课程，如图 4-5 所示所示。

图 4-5　多对多关系图

2. E-R 模型的图示法

E-R 模型可以用一种非常直观的图的形式表示，这种图称为 E-R 图（Entity-Relationship Diagram）。在 E-R 图中我们分别用下面不同的几何图形表示 E-R 模型中的 3 个概念与 2 个连接关系。

（1）实体集表示法。在 E-R 图中用矩形表示实体集，在矩形内写上该实体集的名字。如实体集学生（Student）、课程（Course）可用图 4-6 表示。

（2）属性表示法。在 E-R 图中用椭圆形表示属性，在椭圆形内写上该属性的名称。如学生有属性：学号（S#）、姓名（Sn）及年龄（Sa），它们可以用图 4-7 表示。

（3）联系表示法。在 E-R 图中用菱形（内写上联系名）表示联系。如学生与课程间的联系 SC，用图 4-8 表示。

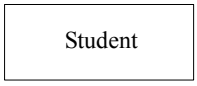


图 4-6　实体集表示法

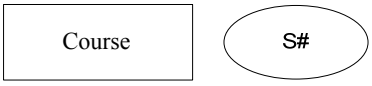


图 4-7　属性表示法

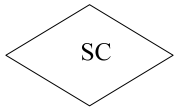


图 4-8　联系表示法

【例题 1】在 E-R 图中，用来表示实体联系的图形是（　　）。（2009 年 9 月）

A）椭圆形　　B）矩形　　C）菱形　　D）三角形

例题分析

实体之间的相互联系，应该用菱形来表示，答案选 C。

4.2.3　关系模型

关系模型采用二维表来表示，简称表。二维表由表框架（Frame）及表的元祖（Tuple）组成。表框架由 n 个命名的属性（Attribute）组成，n 称为属性元数（Arity）。每个属性有一个取值范围称为值域（Domain）。表框架对应了关系的模式，即类型的概念。

1. 关系模型的主要特点

关系数据模型是以集合论中的关系概念为基础发展起来的。关系模型中无论是实体还是实体间的联系均由单一的结构类型——关系来表示。现在流行的数据库都是基于关系模型的，其主要特点如下。

- 一个表中不允许出现相同的两个字段名。
- 一个表中不允许出现完全相同的两行。
- 一个表中同一列的数据项必须是类型相同的数据。

- 一个表中每一行中与每一列中的数据项都是不可拆分的基本数据项。
- 一个表中行或列的顺序改变都不影响表格所描述的内容。

2. 关系模型的重要概念

- 候选码：指的是关系模式中的属性或属性组能唯一区分一条记录。
- 主码（主键）：从候选码中选一个出来作为主码。
- 主属性：是指包含在候选码中的任何属性。
- 非主属性：不包含在任何候选码中的属性。
- 规范化：在关系数据库中每一个关系（数据表，比如学生表、课程表）满足一定的要求，满足不同要求的为不同范式，范式缩写为 NF。
- 1NF：每个分量必须是不可再分的。
- 2NF：每一个非主属性都完全依赖候选码（非主属性不能只依赖候选码中的部分）。
- 3NF：每一个非主属性都不依赖于候选码传递（请注意，这里非主属性都不依赖候选码传递是指非主属性之间不存在传递依赖的关系，至于候选码内是否存在依赖就不用去管了，我们只盯着非主属性）。
- BCNF：每一个确定的属性集都包含候选码（也就是说任何属性对候选码的传递都不行，请注意，这里是说任何属性，也包含主属性，因为在候选码中，也有主属性之间相互传递的情况）。

【例题 2】人员基本信息一般包括：身份证号、姓名、性别、年龄等。其中可以作为主关键字的是_____。

例题分析

由于姓名、性别、年龄这 3 个属性可能存在重复的值，所以无法达到区别一条记录的作用，所以，主关键字为身份证号。

4.2.4 数据操作

关系模型的数据操作是建立在关系上的数据操作，一般有查询、增加、删除及修改 4 种操作。

1. 数据查询

用户可以查询关系数据库中的数据，它包括一个关系内的查询及多个关系间的查询。

（1）对一个关系内查询的基本单位是元组分量，其基本过程是先定位后操作。所谓定位包括纵向定位与横向定位，纵向定位即是指定关系中的一些属性（称列指定），横向定位即是选择满足某些逻辑条件的元组（称为选择）。通过纵向与横向定位后，一个关系中的元组分量即可确定了。在定位后即可进行查询操作，就是将定位的数据从关系数据库中取出并放至指定内存。

（2）对多个关系间的数据查询则可分为 3 步：第一步，将多个关系合并成一个关系；第二步，对合并后的一个关系做定位；第三步，操作。其中第二步与第三步为对一个关系的查询。对多个关系的合并可分解成两个关系的逐步合并。

2. 数据删除

数据删除的基本单位是一个关系内的元组，它的功能是将指定关系内的指定元组删除。它也分为定位与操作两部分，其中定位部分只需要横向定位而无须纵向定位，定位后即执行删除操作。因此数据删除可以分解为一个关系内的元组选择与关系中的元组删除两个基本操作。

3. 数据插入

数据插入仅对一个关系而言，即在指定关系中插入一个或多个元组。在数据插入中不需要定位，仅需做关系中元组插入操作，因此数据插入只有一个基本操作。

4. 数据修改

数据修改是在一个关系中修改指定的元组与属性。数据修改不是一个基本操作，它可以分解为删除需修改的元组与插入修改后的元组两个更基本的操作。

4.2.5　关系中的数据约束

关系模型允许定义 3 类数据约束，它们是实体完整性约束、参照完整性约束及用户定义的完整性约束，其中前两种完整性约束由关系数据库系统自动支持。对于用户定义的完整性约束，则由关系数据库系统提供完整性约束语言，用户利用该语言写出约束条件，运行时由系统自动检查。

（1）实体完整性约束（Entity Integrity Constraint）。该约束要求关系的主键中属性值不能为空值，这是数据库完整性的最基本要求，因为主键是唯一决定元组的，如为空值则其唯一性就成为不可能的了。

（2）参照完整性约束（Reference Integrity Constraint）。该约束是关系之间相关联的基本约束，它不允许关系引用不存在的元组，即在关系中的外键要么是所关联关系中实际存在的元组，要么就为空值。比如在关系 S（S#,SN,SD,SA）与 SC（S#,C#,G）中，SC 中主键为（S#,C#）而外键为 S#，SC 与 S 通过 S#相关联，参照完整性约束要求 SC 中的 S#必在 S 中有相应的元组值，如有 SC（S13,C8,70），则必在 S 中存在（S13,…）。

（3）用户定义的完整性约束（User defined Integrity Constraint）。它是针对具体数据环境与应用环境由用户具体设置的约束，它反映了具体应用中数据的语义要求。

实体完整性约束和参照完整性约束是关系数据库所必须遵守的规则，在任何一个关系数据库管理系统（RDBMS）中均由系统自动支持。

4.3　关系代数

关系代数是一种抽象的查询语言，用来对关系的运算表达查询，作为研究关系数据语言的数学工具。

关系代数的运算对象是关系，运算结果亦为关系。关系代数用到的运算符包括 4 类：集合运算符、专门的关系运算符、比较运算符和逻辑运算符。

比较运算符和逻辑运算符是用来辅助专门的关系运算符进行操作的，所以关系代数的运算按运算符的不同主要分为传统的集合运算和专门的关系运算两类。

传统的集合运算是二目运算，包括并、差、交、广义笛卡儿积 4 种运算。

1. 并（Union）

设关系 R 和关系 S 具有相同的目 n（即两个关系都有 n 个属性），且相应的属性取自同一个域，则关系 R 与关系 S 的并由属于 R 或属于 S 的元组组成。其结果关系仍为 n 目关系。记做：

$$R \cup S=\{t|t\in R \vee t\in S\}$$

2. 差（Difference）

设关系 R 和关系 S 具有相同的目 n，且相应的属性取自同一个域，则关系 R 与关系 S 的差由属于 R 而不属于 S 的所有元组组成。其结果关系仍为 n 目关系。记做：

$$R\text{-}S=\{t|t\in R \wedge t\notin S\}$$

3. 交（Intersection）

设关系 R 和关系 S 具有相同的目 n，且相应的属性取自同一个域，则关系 R 与关系 S 的交由既属于 R 又属于 S 的元组组成。其结果关系仍为 n 目关系。记做：

$$R\cap S=\{t|t\in R \wedge t\in S\}$$

4. 广义笛卡儿积（Extended Cartesian Product）

两个分别为 n 目和 m 目的关系 R 和 S 的广义笛儿尔积是一个 n+m 列的元组的集合。元组的前 n 列是关系 R 的一个元组，后 m 列是关系 S 的一个元组。若 R 有 k_1 个元组，S 有 k_2 个元组，则关系 R 和关系 S 的广义笛卡儿积有 $k_1 \times k_2$ 个元组。

广义的笛卡儿积与并、交、差的不同之处在于，进行交、并、差 3 种运算的时候要求具有相同的列的个数和列名，而进行笛卡儿积的时候则不需要。

关系代数的基本操作介绍如下。

（1）选择。是指从一个关系（表）中找出满足一定条件的所有元组（记录），即在二维表中选取若干行。

（2）投影。是指从一个关系（表）中选择所需要的若干属性（字段），构成一个新的关系（表），即在表中筛选出若干列。

（3）连接。是指从两个关系中按一定的条件分别选取其中的若干属性（字段），形成新的关系。生成的新关系中行的选取，取决于规定的条件。

4.4 数据库设计

数据库设计（Database Design）是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求（信息要求和处理要求）。在数据库领域内，常常把使用数据库的各类系统统称为数据库应用系统。

数据库设计目前一般采用生命周期（Life Cycle）法，即将整个数据库应用系统的开发分解成目标独立的若干阶段。它们是：需求分析阶段、概念设计阶段、逻辑设计阶段、物理设计阶段、编码阶段、测试阶段、运行阶段、维护阶段。

1. 需求分析

数据库的需求分析是调查和分析用户的业务活动和数据的使用情况，弄清所用数据的种类、范围、数量及它们在业务活动中交流的情况，确定用户对数据库系统的使用要求和各种约束条件等，形成用户需求规范。从以下几个方面来进行分析。

（1）技术可行性：主要从项目实施的技术角度，合理设计技术方案，并进行比选和评价。各行业不同项目技术可行性的研究内容及深度差别很大。对于工业项目，可行性研究的技术论证应达到能够比较明确地提出设备清单的深度；对于各种非工业项目，技术方案的论证也应达到目前工程方案初步设计的深度，以便与国际惯例接轨。

（2）经济可行性：主要从资源配置的角度衡量项目的价值，评价项目在实现区域的经济发展目标、有效配置经济资源、增加供应、创造就业、改善环境、提高人民生活等方面的效益。

（3）法律可行性：确认待开发的系统可能会涉及的任何侵犯、妨碍、责任等问题。

2. 概念设计

对用户要求描述的现实世界（可能是一个工厂、一个商场或者一个学校等），通过对其中各处的分类、聚集和概括，建立抽象的概念数据模型。这个概念模型应反映现实世界各部门的信息结构、信息流动情况、信息间的互相制约关系及各部门对信息储存、查询和加工的要求等。所建立的模型应避开数据库在计算机上的具体实现细节，用一种抽象的形式表示出来。以 E-R 模型的联系表示法为例，第一步先明确现实世界各部门所含的各种实体及其属性、实体间的联系及对信息的制约条件等，从而给出各部门内所用信息的局部描述（在数据库中称为用户的局部视图）；第二步再将前面得到的多个用户的局部视图集成为一个全局视图，即用户要描述的现实世界的概念数据模型。

3. 逻辑设计

主要工作是将现实世界的概念数据模型设计成数据库的一种逻辑模式，即适应于某种特定数据库管理系统所支持的逻辑数据模式。与此同时，可能还需要为各种数据处理应用领域产生相应的逻辑子模式。这一步设计的结果就是所谓的“逻辑数据库”。

4. 物理设计

根据特定数据库管理系统所提供的多种存储结构和存取方法等，依赖于具体计算机结构的各项物理设计措施，对具体的应用任务选定最合适的物理存储结构（包括文件类型、索引结构和数据的存放次序与位逻辑等）、存取方法和存取路径等。这一步设计的结果就是所谓的“物理数据库”。

5. 编码设计和测试

在上述设计的基础上，收集数据并具体建立一个数据库，运行一些典型的应用任务来进行数据库的编码，完成后验证

数据库设计的正确性和合理性。一般，一个大型数据库的设计过程往往需要经过多次循环反复测试。当设计的某步发现问题时，可能就需要返回到前面去进行修改。因此，在做上述数据库设计时就应考虑到今后修改设计的可能性和方便性。

6. 运行与维护设计

在数据库系统正式投入运行的过程中，必须不断地对其进行调整和修改。

4.5　习题

4.5.1　选择题

1. 数据库管理系统 DBMS 中用来定义模式、内模式和外模式的语言为（　　）。

A）C　　B）Basic　　C）DDL　　D）DML

2. 数据库设计的 4 个阶段是需求分析、概念设计、逻辑设计和（　　）。

A）编码设计　　B）测试阶段　　C）运行阶段　　D）物理设计

3. 下列描述中错误的是（　　）。

A）在数据库系统中，数据的物理结构必须与逻辑结构一致

B）数据库技术的根本目标是要解决数据的共享问题

C）数据库设计是指在自己有数据库管理系统的基础上建立数据库

D）数据库系统需要操作系统的支持

4. 下列描述中正确的是（　　）。

A）数据库系统是一个独立的系统，不需要操作系统的支持

B）数据库技术的根本目标是要解决数据的共享问题

C）数据库管理系统就是数据库系统

D）以上 3 种说法都不对

5. 设有表示学生选课的 3 张表，学生 S（学号、姓名、性别、年龄、身份证号），课程 C（课号、课名），选课 SC（学号、课号、成绩），则表 SC 的关键字（键或码）为（　　）。

A）课号、成绩　　B）学号、成绩　　C）学号、课号　　D）学号、姓名、成绩

6. 数据库 DB、数据库系统 DBS、数据库管理系统 DBMS 之间的关系是（　　）。

A）DB 包括 DBS 和 DBMS　　B）DBMS 包括 DB 和 DBS

C）DBS 包括 DB 和 DBMS　　D）没有任何关系

7. 如果在一个关系中，存在多个属性（或属性组）都能用来唯一标识关系的元组，且其任何一个子集都不具有这一特性。这些属性（或属性组）都被称为关系的（　　）。

A）连接码　　B）主码　　C）外码　　D）候选码

8. 取出关系中的某些列，并消去重复的元组的关系运算称为（　　）。

A）选择运算　　B）投影运算　　C）连接运算　　D）积运算

9. 将 E-R 图转换到关系模式时，实体与实体间的联系可以表示成（　　）。

A）属性　　B）关系　　C）键　　D）域

10. 在数据库设计过程中，独立于计算机有硬件与 DBMS 软件的设计阶段是（　　）。

A）概念设计　　B）物理设计　　C）逻辑设计　　D）系统实施

11. 数据库的物理设计是为一个给定的逻辑结构选取一个适合应用环境的（　　）的过程，包括确定数据库在物理设备上的存储结构和储存方法。

A）逻辑结构　　B）物理结构　　C）概念结构　　D）层次结构

4.5.2 填空题

1. 数据管理技术的发展经历了人工管理、文件系统和数据库系统 3 个阶段，其中数据独立性最高的是_____。

2. 数据的独立性分为逻辑独立性与物理独立性。当数据的存储结构改变时，其逻辑结构可以不变。基于逻辑结构的应用程序不必修改称为_____。

3. 数据库系统中实现各种数据管理功能的核心软件是_____。

4. 关系数据库系统实现的专门关系运算包括选择、连接和_____。

5. 数据模型按不同应用层次分成 3 种类型，它们是概念数据模型、_____和物理数据模型。

第5章 程序设计基本概念

本章主要介绍了有关程序设计的基本概念及 C 语言程序设计的关键字，结合计算机等级考试要求，具体如表 5-1 所示。

表 5-1　考试要求

考试知识点	重要性
程序设计	★
C 语言关键字	★★

5.1　程序和程序设计

程序是计算机中编写的代码的称呼，而程序设计则是与程序相关的一系列活动的内容。

5.1.1　程序

在计算机中，每个动作、任务都是靠计算机指令来完成的，指令就是计算机的一个特定的基本动作。所以人们编写程序、利用计算机来解决问题，实际上也就是通过计算机按照人的意图有序地执行指令来完成的。程序也就是计算机指令的集合。

5.1.2　程序设计

程序设计是指设计、编制、调试程序的方法和过程。按照结构性质，有结构化程序设计与非结构化程序设计之分。前者是指具有结构性的程序设计方法与过程，它具有由基本结构构成复杂结构的层次性，后者反之。按照用户的要求，有过程式程序设计与非过程式程序设计之分。前者是指使用过程式程序设计语言的程序设计，后者指非过程式程序设计语言的程序设计。程序设计=数据结构+算法。

5.1.3　程序设计语言

程序设计语言是用于编写计算机程序的语言。语言的基础是一组记号和一组规则。根据规则由记号构成的记号串的总体就是语言。在程序设计语言中，这些记号串就是程序。程序设计语言包含 3 个方面，即语法、语义和语用。语法表示程序的结构或形式，亦即表示构成程序的各个记号之间的组合规则，但不涉及这些记号的特定含义，也不涉及使用者；语义表示程序的含义，亦即表示按照各种方法所表示的各个记号的特定含义，但也不涉及使用者；语用表示程序与使用的关系。

5.2 C 语言的语句和关键字

C 语言就是其中一种程序设计语言，C 语言由一个主函数和若干个子函数组成。

5.2.1 C 程序的基本结构

```
#include <stdio.h>
void  main()
{
    printf("欢迎进入 C 语言学习！\n");            //输出语句
}
```

说明

（1）#include，以#开始的语句称为预处理行，也是命令行，在源程序编译之前对这些命令进行处理，处理结果和源程序一起进行编译。并不是任何程序都需要预处理行的，位置必须是程序的开始处。

（2）<stdio.h>，以.h 为扩展名的文件称为头文件，一般是 C 语言中的标准库文件，亦可以是用户自定义的库文件。这些标准的库文件就是为了程序设计者方便开发，预先编写好的资源文件，例如，输入输出的函数资源文件就包含在 stdio.h（Standards input & output）中，这些库文件可以在程序内任意调用。

（3）void main()，main(){}这样的结构是一个函数，C 语言程序是由一个或者多个函数构成的，其中 main()是一个函数的头部，该函数就是主函数，也是程序的入口，开始执行的地方。函数就是一个完成特定功能的独立处理模块，函数可以有返回值（处理结果），也可以没有返回值，没有返回值的函数名前加一个关键字 void。

（4）{}，是表示函数体的限定符号，函数体就是完成特定功能的处理代码。{}分别表示函数体的开始和结束。

（5）“printf("欢迎进入 C 语言学习！\n");”称为 C 语言中的语句，每一条语句都是以“;”号结束，也就是函数体内的语句，作用是在屏幕上显示“欢迎进入 C 语言学习！”，\n 表示换行，printf()就是包含在 stdio.h 头文件中的一个输出函数。

（6）//，表示单行的注释，多行注释可以用/*与*/。注释为了便于阅读、理解、维护程序，所以不能忽视程序注释的作用。

（7）用{}括起来的部分，一般是一定层次的语句块，最好一个{和}独占一行，低一层次的代码要与高一层次的代码有缩进，这样方便阅读程序，增加层次结构感。这些都是编程的良好习惯，在以后的编程中，希望大家能慢慢养成好的编程习惯。

5.2.2 C 语言语句

由于 C 语言的组成部分是函数，函数大部分由语句组成，C 语言的语句由分号结束。C 程序的执行部分是由语句组成的，程序的功能也是由执行语句实现的。

C 语句可分为以下 5 类。

1. 表达式语句

表达式语句由表达式加上分号组成，其一般形式为“表达式;”。执行表达式语句就是计算表达式的值。例如，“x=y+z;”是赋值语句，“y+z;”是加法运算语句，但计算结果不能保留，无实际意义；“i++;”是自增 1 语句，i 值增 1。

2. 函数调用语句

由函数名、实际参数加上分号组成。其一般形式为“函数名（实际参数表）;”，执行函数语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值。例如，“printf("C Program");”调用库函数，输出字符串。

3. 控制语句

控制语句用于控制程序的流程，以实现程序的各种结构方式，它们由特定的语句定义符组成。C 语言有 9 种控制语句，可分成以下 3 类。

（1）条件判断语句

if 语句，switch 语句。

（2）循环执行语句

do while 语句，while 语句，for 语句。

（3）转向语句

break 语句，goto 语句，continue 语句，return 语句。

4. 复合语句

把多个语句用花括号{}括起来组成的一个语句称复合语句，在程序中应把复合语句看成是单条语句，而不是多条语句，例如：

```
{
   x=y+z;
   a=b+c;
   printf("%d%d",x,a);
}
```

是一条复合语句。复合语句内的各条语句都必须以分号结尾，在花括号}外不能加分号。

5. 空语句

只由分号组成的语句称为空语句。空语句是什么也不执行的语句。在程序中空语句可用来做空循环体。例如“while(getchar()!='\n');”语句的功能是，只要从键盘输入的字符不是回车则重新输入，这里的循环体为空语句。

6. 赋值语句

赋值语句是由赋值表达式再加上分号构成的表达式语句。其一般形式为“变量=表达式;”，赋值语句的功能和特点都与赋值表达式相同，它是程序中使用最多的语句之一。在赋值语句的使用中需要注意以下几点。

（1）由于在赋值符=右边的表达式也可以是一个赋值表达式，因此，下述形式“变量=（变量=表达式);”是成立的，从而形成嵌套的情形。其展开之后的一般形式为“变量=变量=……=表达式;”。

例如：

```
a=b=c=d=e=5;
```

按照赋值运算符的右接合性，实际上等效于：

```
e=5;
d=e;
c=d;
b=c;
a=b;
```

（2）注意在变量说明中给变量赋初值和赋值语句的区别。给变量赋初值是变量说明的一部分，赋初值后的变量与其后的其他同类变量之间仍必须用逗号间隔，而赋值语句则必须用分号结尾。

（3）在变量说明中，不允许连续给多个变量赋初值。如下述说明是错误的：“int a=b=c=5”必须写为“int a=5,b=5,c=5;”，而赋值语句允许连续赋值。

（4）注意赋值表达式和赋值语句的区别。赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句则不能。

下述语句是合法的：“if((x=y+5)>0) z=x;”，语句的功能是，若表达式 x=y+5 大于 0 则 z=x。下述语句是非法的：“if((x=y+5;)>0) z=x;”，因为“x=y+5;”是语句，不能出现在表达式中。

7. 数据输出语句

本小节介绍的是向标准输出设备显示器输出数据的语句。在C语言中，所有的数据输入/输出都是由库函数完成的，因此都是函数语句。本小节先介绍 printf 函数和 putchar 函数。printf 函数称为格式输出函数，其关键字最末一个字母 f 即为“格式”（Format）之意。其功能是按用户指定的格式，把指定的数据显示到显示器屏幕上。

5.2.3 关键字

C 语言简洁、紧凑，使用方便、灵活。ANSI C 一共有 32 个关键字。

auto break case char const continue default
do double else enum extern float for
goto if int long register return short
signed static sizeof struct switch typedef union
unsigned void volatile while

5.3 习题

5.3.1 选择题

1. C 语言可执行程序的开始执行点是（　　）。

A）包含文件中的第一个函数　　B）程序中第一个函数
C）程序中的 main()函数　　D）程序中第一条语句

2. 下列叙述中错误的是（　　）。

A）一个 C 语言程序只能实现一种算法
B）C 程序可以由多个程序文件组成
C）C 程序可以由一个或多个函数组成
D）一个函数可以单独作为一个 C 程序文件存在

3. 下列叙述中错误的是（　　）。

A）计算机不能直接执行用 C 语言编写的源程序
B）C 程序经 C 编译后，生成的后缀为.obj 的文件是一个二进制文件
C）后缀为.obj 的文件，经连接程序生成的后缀为.exe 的文件是一个二进制文件
D）后缀为.obj 的二进制文件都可以直接运行

4. C 语言源程序名的后缀是（　　）。

A）.exe　　B）.c　　C）.obj　　D）.cp

5. 下列叙述中错误的是（　　）。

A）C 语言区分大小写
B）C 程序中的一个变量，代表内存中一个相应的存储单元，变量的值可以根据需要随时修改
C）整数和实数都能用 C 语言准确无误地表示出来
D）在 C 程序中，正整数可以用八进制、十进制和十六进制的形式来表示

6. C 语言规定，在一个 C 程序中，main()函数的位置（　　）。

A）必须在系统调用的库函数之后　　B）必须在程序的开始
C）必须在程序的最后　　D）可以在任意位置

7.下列叙述中正确的是（　　）。

A）main()函数必须在 C 程序的最后面

B）main()函数必须在 C 程序的前面

C）main()函数必须在 C 程序的中间部分，但在执行 C 程序时是从程序开头开始的

D）main()函数可以在 C 程序的中间部分，但在执行 C 程序时是从 main 函数开始的

8. C 语言的基本单位是（　　）。

A）函数　　B）过程　　C）子程序　　D）子函数

9. 下列叙述中正确的是（　　）。

A）C 程度由函数组成

B）C 程序由主函数构成

C）C 程序由函数和过程构成

D）在 C 语言中无论是整数还是实数，都可以正确无误地表示出来

10. 下列说法正确的是（　　）。

A）在 C 程序中，主函数必须在主程序的最前面

B）C 程序的书写格式是固定的，每行只能写一条语句

C）C 语言主函数 main 后的括号是不可以省略的

D）C 语言的主函数必须位于第一行

5.3.2　填空题

1. C 语言用于结构化程序设计的 3 种基本结构是_____、选择结构和循环结构。

2. 复合语句在语法上被认为是_____。空语句的形式是_____。

3. C 语句句尾用_____结束。

第6章 C语言数据类型、运算符和表达式

本章要学习的主要内容是C语言的数据类型概念，变量和常量的概念，整型、实型、字符型等基本数据类型的用法，以及算术运算、赋值运算、逗号运算的运算符和运算规则及表达式，以及不同类型的类型转换。结合计算机等级考试要求，具体如表6-1所示。

表6-1　考试要求

考试知识点	重要性
数据类型概念	★
基本数据类型常量	★★
基本数据类型变量	★★★
算术运算符及表达式	★★
逗号运算符、赋值运算符	★★
类型转换	★

6.1　C语言数据类型

在C语言中，数据类型可分为：基本数据类型、构造数据类型、指针类型、空类型4大类。

1. 基本数据类型

基本数据类型最主要的特点是，其值不可以再分解为其他类型，也就是说，基本数据类型是自我说明的。

2. 构造数据类型

构造数据类型是根据已定义的一个或多个数据类型用构造的方法来定义的。也就是说，一个构造类型的值可以分解成若干个“成员”或“元素”。每个“成员”都是一个基本数据类型或又是一个构造类型。在C语言中，构造类型有以下几种。

（1）数组类型。

（2）结构类型。

（3）联合类型。

数据类型对于不同类别的数据进行分类，目的是为了存储。在程序编写的过程中，任何一种数据都要指定类型，C语言的数据类型，如图6-1和图6-2所示。

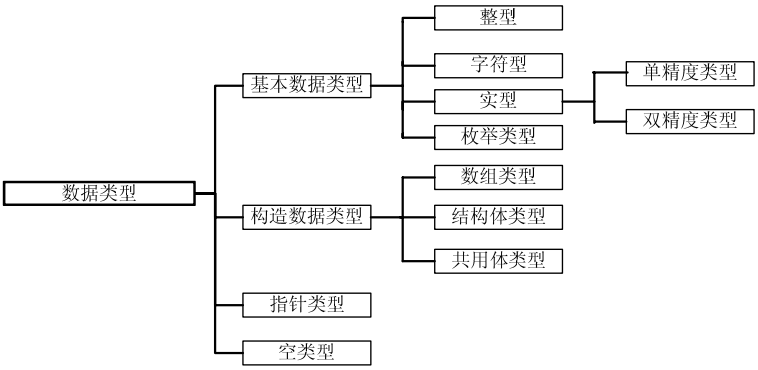


图 6-1　C 语言的数据类型

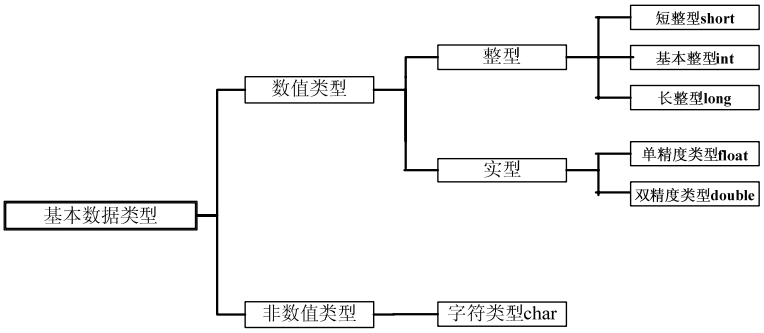


图 6-2　C 语言的基本数据类型

6.2　常量、变量和标识符

计算机中程序有的时候需要运行数据，而这些数据有的是固定不变的，有的是会发生变化的，那下面就具体看看这些数据是如何表示的。

6.2.1　常量

所谓常量是指在整个程序运行的过程中，其值固定不变的量。常量有以下几个特点。

（1）定义时一般取名为全部大写，定义的时候必须有一个值。一个符号常量 PI 的定义如：#define PI 3.14159。

（2）在程序中通常要频繁地使用这个常量。

6.2.2　变量

所谓变量是指在程序运行过程中其值可以改变的量。程序中用到的所有变量都必须有一个名字作为标识，变量的名字由用户定义，它必须符合标识符的命名规则。

一个变量实质上代表了内存中的某个存储单元。在程序中，变量 a 就是指用 a 命名的某个存储单元，用户对变量 a 进行的操作就是对该存储单元进行的操作；给变量 a 赋值，实质上就是把数据存入该变量所代表的存储单元中。

C 语言规定程序中的所有变量都必须先定义后使用。对变量的定义通常放在函数体内的前部，但也可以放在函数的外部或复合语句的开头。

在书写变量说明时，应注意以下几点。

（1）允许在一个类型说明符后，说明多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至

少用一个空格间隔。

（2）最后一个变量名之后必须以“;”符号结尾。

（3）变量说明必须放在变量使用之前。一般放在函数体的开头部分。

6.2.3 标识符

变量的名字有一定的规则，而这个规则也就是标识符，标识符规定如下。

（1）可以由大小写字母（A~Z 或者 a~z）、数字（0~9）、下画线（_）组成。

（2）首个字符必须是字母或者下画线。

（3）不能和 C 语言中的关键字相同。

（4）有大小之分。

【例题 1】以下选项中合法的标识符是（　　）。（2009 年 3 月）

A）l_l　　B）l-l　　C）_ll　　D）l_ _

例题分析

此题考查合法标识符。标识符只能由数字、字母、下画线组成，必须以字母或下画线开头，所以应该选 C 。

【例题 2】以下选项中，能用做用户标识符的是（　　）。（2009 年 9 月）

A）void　　B）8_8　　C）_0_　　D）unsigned

例题分析

根据标识符的定义，此题选 C，A、D 选项都是关键字，而选项 B 是以数字开头的。

6.3 整型数据

计算机中使用整型数据来参与某些程序的运算。

6.3.1 整型常量的表示

整型常量有 3 种表示形式，即十进制（Decimal）、八进制（Octal）、十六进制（Hex）。

（1）十进制整数。如 21，-45，0。

（2）八进制整数。这类常量是以数字 0 开头的，其码值为 0~7，如 012，076。所以，C 语言程序不能在一个十进制数前随意添加“0”，以免引起混淆。

（3）十六进制整数。这类常量是以数字 0 和字母 x 结合开头的，其码值为 0~9 和 A~F（或者 a~f），如 0xaf，-0x23。

在 C 语言中，只有十进制数可以是负数，而八进制和十六进制只能是正数。

6.3.2 整型变量

在 C 语言中，整型数据可以分为 3 种类型，即短整型（short int）、基本整型（int）、长整型（long int），不同的数据类型分配不同的长度存储。整型数据又可以分为有符号类型和无符号类型两种，默认的整型数据是有符号类型的。整型数据 3 种类型任何一种需要做无符号处理的，都可以在定义前加关键字 unsigned，如表 6-2 所示。

表 6-2　整型数据

名称	变量说明	省略说明	所占位数	表示范围
短整型	short int	short	16 位（2 字节）	-32767~32768
基本整型	int	int	16 位（2 字节）	-32767~32768
长整型	long int	long	32 位（4 字节）	-2147483648 ~2147483647

（续表）

名称	变量说明	省略说明	所占位数	表示范围
无符号短整型	unsigned short int	unsigned short	16 位（2 字节）	0~65535
无符号整型	unsigned int	unsigned	16 位（2 字节）	0~65535
无符号长整型	unsigned long int	unsigned long	32 位（4 字节）	0~4294967295

6.3.3　整数在内存中的存储形式

在计算机中，内存储器的最小存储单位为“位（bit）”。由于只能存放 0 或 1，因此称为二进制位。大多数计算机把 8 个二进制位组成一个“字节（byte）”，并给每个字节分配一个地址。每一种操作系统存放计算机指令的字节数是不一样的，有的计算机用两个字节（16 个二进制位）来存放一条机器指令，称此计算机的字长为 16 位；有的计算机用 4 个字节（32 个二进制位）来存放一条机器指令，称此计算机的字长为 32 位。

通常把一个字节中的最右边一位称为最低位，最左边一位称为最高位。对于一个有符号整数，其中最高位（最左边的一位）用来存放整数的符号，称为符号位。若是正整数，最高位放置 0；若是负整数，最高位放置 1。

1. 正整数

当用两个字节存放一个正整数时，例如，正整数 7，其在内存中的二进制码为：

0000000000000111

对于正整数的这种存储形式称为用“原码”形式存放。因此用两个字节存放 short 类型的最大正整数：

0111111111111111

它的值为 32767。

2. 负整数

计算机中的正整数是以二进制的形式存在的，那负整数该以什么样的方式存储呢？在计算机中为了区分正数和负数，该数的绝对值变成二进制，二进制的第一位取 0，表示正数，取 1，表示负数。

（1）负整数在内存中是以“补码”形式存放的。

取某个二进制数的补码，例如，10000111（十进制数-7）的补码，步骤如下。

- 求原码的反码。把原码除符号位之外的二进制码按位取反，即把 1 变成 0，0 变成 1，得到该原码的反码。例如，10000111 的反码为 11111000。
- 把所得的反码加 1，即得到原码的补码。因此 11111000 加 1 得到 11111001，这就是-7 在内存中的二进制码。若用两个字节表示，即为 11111001。

（2）把内存中以补码形式存放的二进制码转化成十进制的负整数，步骤如下。

- 先对除符号位之外的各位取反。例如，有补码 11111001，取反后为 10000110。
- 将所得二进制数转换为十进制数。例如，10000101 的十进制数为-6。
- 对所求得的数再减 1，即为-7。

3. 无符号整数

用两个字节存放一个整数时，若有说明为无符号整数，则高位不再用来存放整数的符号，16 个二进制位全部用来存放整数，因此无符号整数不可能是负数。这时，若内存中存放的 16 个二进制位全部为 1，则它所代表的整数就不再是-1，而是 65535。

6.3.4　常用的输出格式

%d 按整型数据的实际长度输出。

%md 使输出长度为 m，如果数据长度小于 m，则左补空格；如果大于 m，则输出实际长度。

%ld 输出长整型数据。

6.4 实型数据

在计算机中使用实型数据来参与某些程序的运算。

6.4.1 实型常量的表示方法

实型常量可以分为两种表示形式。

（1）小数表示形式，即小数点和数字的表示，如：.123、135.34、0.0 等。

（2）指数表示形式，1.2E3、4.3e-2 等。这里要说明的是 E 或者 e 表示基数 10，之前必须是一个小数，之后必须是一个整数。

【注意】

（1）可以在一个常量之后加 L（或者 l）表示长双精度类型。

（2）浮点常数默认存储为 double 类型。

6.4.2 实型变量

实型变量可以分为单精度（float）、双精度（double）和长双精度（long double）类型，如表 6-3 所示。

表 6-3 实型变量

名称	变量说明	有效数字	所占位数	表示范围
单精度	float	6~7 位	32 位（2 字节）	10^{-37}~10^{38}
双精度	double	15~16 位	64 位（2 字节）	10^{-307}~10^{308}
长双精度	long double	18~19 位	128 位（4 字节）	10^{-4931}~10^{4932}

6.4.3 常用的输出格式

%f 整数部分全部输出，小数部分输出 6 位。

%m.nf 输出数据共占 m 位，其中有 n 位小数。如果数值长度小于 m，左补空格。

%-m.nf 同上，右补空格。

6.5 算术表达式

C 语言功能强大，灵活性很高，不但提供了丰富的数据类型，也提供了大量的运算符，运算符和数值结合起来做运算得出结果。表达式是由常量、变量、函数和运算符组合起来的式子。一个表达式有一个值及其类型，它们等于计算表达式所得结果的值和类型。表达式求值按运算符的优先级和结合性规定的顺序进行，单个常量、变量、函数可以看做是表达式的特例。

算术表达式是用算术运算符和括号将运算对象（也称操作数）连接起来的、符合 C 语法规则的式子。

6.5.1 算术运算符

基本的算术运算符如表 6-4 所示。

表 6-4　算术运算符

算术运算名称	运算符	说明	表达式举例
加法	+	加法运算	num1 + num2
减法	-	减法运算	num1 - num2
乘法	*	乘法运算	num1 * num2
除法/整除	/	除法运算或者整除运算	num1 / num2
取余数	%	整除后的余数	num1 % num2
自加	++	操作数增加 1	num++或者++num
自减	--	操作数减少 1	num--或者--num

算术运算又可以分单目运算符和双目运算符。

（1）单目运算符。单目运算符就是指那些运算处理只有一个操作数的运算符。在算术运算符中有++、--两种运算符是单目元算符。

（2）双目运算符。双目运算符是指运算符有两个操作数的，算术运算符中的双目运算符有+，-，*，/，%。

/是除法运算或整除运算，当两个整数相除就是整除运算，其他都为带小数的除法。如果整数需要做带小数的除法时，可以给整数后加.0，或者进行类型的强制转换，如 2/3 就是做整除运算，结果为 0，而 2.0/3.0 这样的结果就是 0.677777。在做整除运算时，保证正确性，两个操作数不能有负整数。

%求余数运算，需要参与的操作数必须是整数。如，2%3=2，5%4=1。

以上所列的运算符中，只有单目运算符+和-的结合性是从右到左的，其余运算符的结合性都是从左到右的。但是如果遇到括号的话，那么优先级的顺序就可能会改变。

例如，表达式(4+3)%4 的运算结果是 3，圆括号的优先级高于取余号；表达 4+3%4 的运算结果是 7，取余号的优先级高于加号。可见在算术表达式中，使用括号的结果是不一样的。

6.5.2　算术表达式

用算术运算符和一对圆括号将运算数（或称操作数）连接起来的、符合 C 语言语法的表达式称为算术表达式。

在算术表达式中，运算对象可以是常量、变量和函数等。例如，sqrt(a)+fabs(b)。

在计算机语言中，算术表达式中的运算规则和要求如下。

（1）在算术表达式中，可使用多层圆括号，但左右括号必须配对。运算时从内层圆括号开始，由内向外依次计算表达式的值。

（2）在算术表达式中，若包含不同优先级的运算符，则按运算符的优先级由高到低进行；若表达式中运算符的级别相同，则按运算符的结合方向进行。例如，表达式 a+b-c，因为+号和-号的优先级相同，它们的结合性为从左到右，因此先计算 a+b，然后把所得结果减去 c 的值。

强制类型转换表达式的形式如下。

（类型名）（表达式）

在上述形式中，（类型名）称为强制类型转换运算符，利用强制类型转化运算符可以将一个表达式的值转换成指定的类型，这种转换是根据人为要求进行的。例如，表达式“(int)3.234”把 3.234 转换成整数 3，表达式(double)(10%3)把 10%3 所得结果 1 转换成双精度数。

6.6　赋值表达式

赋值表达式在 C 语言中具有重要的地位，通过它可以使变量的值发生改变。

6.6.1 赋值运算符和赋值表达式

1. 赋值运算符和赋值表达式

在 C 语言中，赋值号=是一个运算符，称为赋值运算符。由赋值运算符组成的表达式称为赋值表达式，其形式如下：

变量名=表达式

赋值号的左边必须是一个表达某一存储单元的变量名，赋值号的右边必须是 C 语言中合法的表达式。赋值运算的功能是先求出右边表达式的值，然后把此值赋给赋值号左边的变量。

说明：

（1）x=3。表示将 3 的值赋值给变量 x，即 x 的值是 3。

（2）赋值表达式 x=y 的作用是，将变量 y 所代表的存储单元中的内容赋给变量 x 所代表的存储单元，x 中原有的数据被替换掉。赋值后，变量 y 中的内容保持不变。

（3）表达式 m=m+1 也是合法的赋值表达式，其作用是取变量 m 中的值加 1 后再放回到变量 m 中，即使变量 m 中的值增 1。

（4）赋值运算符的左侧只能是变量，不能是常量或表达式。a+b=c 就是非法的赋值表达式。

（5）等号右边的表达式也可以是一个赋值表达式。如 a=b=2+3，按照运算符的优先级，将首先计算出 2+3 的值 5，然后按照赋值运算符自右向左的结合性，把 5 赋给变量 b，最后再把变量 b 的值赋给变量 a。

（6）当赋值号左边的变量为短整型，右边的值为长整型时，短整型变量只能接受长整型数低位上两个字节中的数据，高位上两个字节中的数据将丢失。也就是说，右边的值不能超出短整型的数值范围，否则将得不到预期的结果。

（7）当赋值号左边的变量为无符号整型，右边的值为有符号整型时，则把内存中的内容原样复制。右边数值的范围不应超出左边变量可以接受的数值范围。同时需要注意，这时负数将转换为整数。

（8）当赋值号左边的变量为有符号整型，右边的值为无符号整型时，复制的机制同上。这时若符号位为 1，将按负数处理。

【考生注意】

赋值运算符和后面所讲的关系运算符中的= =很容易混淆，要特别注意。

在程序中可以多次给一个变量赋值，每赋一次值，与它相应的存储单元中的数据就被更新一次，内存中该数据就是最后一次所赋的那个数据。

例如：int a=3;

```
a=a+2;
```

第一次变量 a 的值为 3，经过 a=a+2 之后，值变为了 5。可见第二次对 a 所赋的值已经发生了改变，把第一次的 a 值覆盖掉了，只剩下最近一次 a 的值。

2. 复合赋值表达式

在赋值运算符之前加上其他运算符可以构成复合赋值运算符。C 语言规定可以使用 10 种复合赋值运算符，其中与算术运算有关的复合赋值运算符有：+ =，- =，* =，/ =，% =（注意：两个符号之间不可以有空格）。复合赋值运算符的优先级与赋值运算符的优先级相同。表达式“n+=1”的运算规则等价于“n=n+1”，表达式“n*=m+3”的运算规则等价于“n=n*(m+3)”，因为运算符“+”的优先级高于复合赋值运算符“*=”。其他以此类推。

【例题 3】已有变量 a=3，计算表达式 a-=a+a 的值。

例题分析

因为赋值运算符与复合赋值运算符-=和+=的优先级相同，且运算方向自右至左，所以：

（1）先计算“a+a”，因 a 的初值为 3，所以该表达式的值为 6，注意 a 的值未变。

（2）再计算“a-=6”，此式相当于“a=a-6”，因 a 的值仍为 3，所以表达式的值为-3，注意 a 的值已为-3。

由此可知，表达式a-=a+a的值是-3。

6.6.2 不同类型数据间的混合运算

在赋值运算中，只有在赋值号右侧表达式的类型与左侧变量类型完全一致时，赋值操作才能进行。如果赋值运算符

两侧的数据类型不一致，在没赋值前，系统将自动先把右侧表达式求得的数值按赋值号左边变量的类型进行转换，也可以用前面强制类型转换的方式人为地进行转换后将值赋给赋值号左边的变量。特别需要指出的是在进行混合运算时，整型数据类型之间的转换问题。在 C 语言的表达式（不包括赋值表达式）中，如果运算符两边的整数类型不相同，将进行类型之间的转换，转换规则如下。

整型、字符型、实型数据间可以混合运算，运算时不同类型数据要转换成同一类型再运算，转换规则如图 6-3 所示。

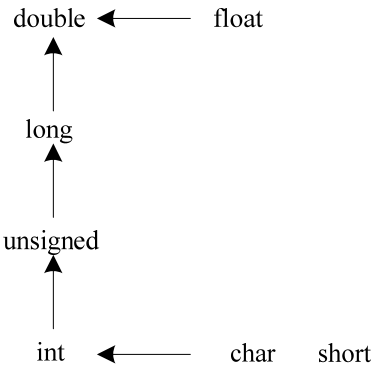


图 6-3　类型转换图

char,short -> int -> unsigned -> long -> double <- float

【例题 4】以下表达式结果的类型是什么？

4+'c'+6.2

例题分析

此题中有 3 种类型的数据，分别是整型、字符型和浮点型中的单精度型，根据以上转换原则，得到的结果是单精度数据。

【例题 5】　(int)((double)9/2)-(9)%2 的值是（　　）。（2009 年 9 月）

A）0

B）3

C）4

D）5

例题分析

此题首先执行(double) 9/2，得到结果 4.5，然后再执行(int) 4.5 得到 4，而后面(9) %2 的结果是 1，所以 4-1=3。答案选 B。

6.7　自增、自减运算和逗号表达式

自增和自减在 C 语言的运算中占有比较重要的位置，它通过对自己的运算，使变量的值增加 1 或减少 1。

6.7.1　自增、自减运算

++、--运算符的操作数只能是一个变量，++的运算最终结果是使其操作数的值增加 1，--的运算最终结果是使其操作数的值减少 1，但是对于这两种运算符所构成的表达式的值的计算是有所区别的，主要区别在于这两种运算符所构成的表达式的形式不同，如表 6-5 所示。

表 6-5　运算符说明

形式	类型	表达式举例 （n 的初值是 0）	m 的值 （表达式的值）	n 的值
++n	运算符前置	m=++n	1	1
--n		m=--n	-1	-1
n++	运算符后置	m=n++	0	1
n--		m=n--	0	-1

从上表中可以看出，当“运算符前置”的时候是“先计算后使用”，当“运算符后置”的时候是“先使用后计算”。这里的计算指的是增加 1，或者减少 1，而这里的使用指的是使用和++、--运算符组成表达式的操作数的值，这里的主要区别是使用操作数变化前或后的值。

【例题 6】写出下列程序的结果。

```
#include "stdio.h"
void main()
{
     int m=5,n,y;
     n=--m;
     y=m++;
     printf("n=%d,y=%d,m=%d",n,y,m);
}
```

例题分析

首先对 n=--m 的式子进行计算，对 m 进行减 1 的操作，得到 n 的值 4，然后再执行 y=m++，在执行之前，要注意的是此时的 m 值不是原来的 5 了，而变成了 4，因为刚刚对 m 进行了减 1 的操作，所以，y 的值是 4，而 m 进行了加 1 操作后变成了 5。所以，结果输出 n=4，y=4，m=5。

【考生注意】遇到这种题目的时候，一定要理解好自增和自减的含义，并且注意语句之间的关系。

6.7.2　逗号表达式

在 C 语言中逗号“,”也是一种运算符，称为逗号运算符。其功能是把两个表达式连接起来组成一个表达式，称为逗号表达式。

其一般形式为：

表达式 1，表达式 2，表达式 3，……，表达式 n

其求值过程是分别先后依次求出 n 个表达式的值，并以最后一个表达式 n 的值作为整个逗号表达式的值。

【例题 7】逗号运算程序举例。

```
#include <stdio.h>
void main()
{
   int a=2,b=4,c=6,y;
   y=(a+b,b+c);
   printf("y=%d",y);
}
```

例题分析

此题是一个逗号表达式的应用，很显然 y 的值就是 b+c 的值即 10。

6.8　习题

6.8.1　选择题

1. 若变量均已正确定义并赋值，以下合法的 C 语言赋值语句是（　　）。

A）x=y==5;　　B）x=n%2.5;　　C）x+n=i;　　D）x=5=4+1;

2. 可在 C 程序中用做用户标识符的一组标识符是（　　）。

A) and　　B）Date　　C）Hi　　D）case

_2007　　y-m-d　　Dr.Tom　　Big 1

3. 若 a 为 int 类型，且其值为 3，则执行完表达式 a+=a-=a*a 后，a 的值是（　　）。

A）-3　　B）9　　C）-12　　D）6

4. 以下选项中，当 x 为大于 1 的奇数时，值为 0 的表达式为（　　）。

A）x%2==1　　B）x/2　　C）x%2!=0　　D）x%2==0

5. 可以在 C 语言程序中用做用户标识符的一组标识符是（　　）。

A）void　　B）aa

123　　_abc

BBN　　cas

C）as+b3　　D）6f

-123　　do

if　　SIG

6. 若函数中有定义语句“int k;”，则（　　）。

A）系统将自动给 k 赋初值 0　　B）这时 k 中的值无定义

C）系统将自动给 k 赋初值-1　　D）这时 k 中无任何值

7. 以下选项中正确的定义语句是（　　）。

A）double a;b　　B）double a=b=7;　　C）double a=7,b=7;　　D）double,a,b;

8. 设变量已正确定义并赋值，以下正确的表达式是（　　）。

A）x=y*5=x+z　　B）int(15.8%5)　　C）x=y+z+5,++y　　D）x=25%5.0

6.8.2　填空题

1. 设变量已正确定义为整型，则表达式 i=2,++i,i++的值为_____。

2. 已知字母 A 的 ASCII 码为 65。以下程序运行后的输出结果是_____。

```
#include <stdio.h>
main()
{
   char a,b;
    a='A'+'5'-'3';    b=a+'6' -'2';
    printf("%d %c\n",a,b);
}
```

3. 执行以下程序后的输出结果是_____。

```
#include <stdio.h>
main()
```

```
{
    int a=10;
     a=(3*5,a+4);
     printf("a=%d\n",a);
}
```

4. 以下程序运行后的输出结果是_____。

```
#include <stdio.h>
main()
{
   int m=012,n=11;
    printf("%d  %d\n",++m,n++);
}
```

5. 已知“int a=2,b=2,c=3;”，则执行完语句“a*=18+(b++)-(++c);”后，a 的值是_____。

第7章 顺序结构程序设计

本章主要学习 C 语言中的输入/输出函数，格式输入/输出的函数是 scanf()和 printf()，单个字符的输入/输出函数 getchar()和 putchar()，不同数据类型的输入/输出方法，以及简单的顺序程序设计，结合计算机等级考试二级 C 语言要求，具体如表 7-1 所示。

表 7-1　考试要求

考试知识点	重要性
格式输入/输出函数	★★★
字符输入/输出函数	★★★

7.1　格式化输出 printf()函数

输出函数是将 C 语言程序的结果输出到屏幕上，通过此函数可以看到程序运行的结果。

7.1.1　基本格式

printf()函数是 C 语言中最基本的数据输出函数，其作用是将内存中的数据按照指定的格式输出到终端设备上（如显示器）。这里指定的格式是通过格式符来实现的，其一般形式为：

printf（“格式描述串”，变量列表）；

（1）“格式描述串”是用双引号引起来的，作用是采用什么样的格式输出变量，主要分类是根据数据的不同类型来划分的。格式描述串中可以包含转换字符字符串（以“%”开头），也可以包含转义字符序列（以“\”标识）。

（2）变量列表，也就是要输出显示的参数列表，多个变量参数之间用“,”号分隔。

（3）格式描述串中也可以没有以“%”开头的描述符，这时“ ”号之内的内容是原样输出的（转义字符序列按照转义字符含义输出），这时，不需要之后的“,”分隔符，当然也不存在变量列表了。

7.1.2　格式说明

1. 格式描述串

格式描述串用于指定输出格式。格式描述串中出现的格式描述符必须和输出变量列表中的变量个数、类型一一对应。格式描述符用以说明输出数据的类型、形式、长度、小数位数等。其格式描述串的一般形式为：

%[对齐标志][输出所占最小列宽度][.小数精度][长短类型]类型

[]表示可选项，其中各项的具体含义如下。

（1）%：格式描述符的开头标志。

（2）类型：类型字符用以表示输出数据的类型，如表 7-2 所示。对齐格式如表 7-3 所示。

表 7-2　输出格式串的类型

数据的类型	格式字符	意　义
整型	d	以十进制形式输出带符号整数（正数不输出符号）
	o	以八进制形式输出无符号整数（不输出前缀 0）
	x,X	以十六进制形式输出无符号整数（不输出前缀 0x）
	u	以十进制形式输出无符号整数
实型	f	以小数形式输出单、双精度实数
	e,E	以指数形式输出单、双精度实数
	g,G	以%f 或%e 中较短的输出宽度输出单、双精度实数
字符型	c	输出单个字符
字符串	s	输出字符串

表 7-3　对齐格式

标 志	意　义
-	输出结果左对齐，右边补空格
+	输出符号（正号或负号），输出结果右对齐，左边补空格

2. 格式描述串的作用

（1）给输出项提供输出格式说明

输出格式说明的作用是将要输出的数据按照指定的格式输出。格式说明由“%”符号和紧跟在其后的格式描述符组成。当输出项为 int 类型时，用 d 作为格式描述字符，其形式为%d；当输出项为 float 或 double 类型时，用 f 或 e 作为格式描述字符，其形式为%f 或%e（对于 double 类型也可用%lf 或%le）；当输出项为 char 类型时，用 c 作为格式描述字符，其形式为%c。

（2）提供需要原样输出的文字或字符

除了格式转换说明外，字符串中的其他字符（包括空格）将按原样输出。例如，“a=%d,b=%d”,其中的“a=”、“,”和“b=”都将原样输出，这样使得输出结果更具可读性。若 a，b 的值分别为 3 和 4，则 printf 输出的结果是“a=3,b=4”。

（3）其他的一些输出格式

输出项可以是任意合法的常量、变量或表达式，也可以没有输出项，函数的调用形式将为 printf（格式控制），输出结果就是格式控制中的固定字符串。比如“printf("OK!");”将输出字符串 OK!。

对于 printf 函数的调用形式，请见下面的程序示例。

```
#include <stdio.h>
main()
{
   int a=3;
   char b='A';
   double c=3.1415;
   printf("a=%d,b=%c,c=%f\n",a,b,c);
}
```

运行后的输出结果为：

a=3,b=A,c=3.1415

说明：

在以上 printf 的输出格式控制中，“a=”按原样输出，在%d 的位置上输出变量 a 的值，接着输出一个逗号和“b=”，

在%c 的位置上输出变量 b 的值，又输出一个逗号和“c=”，在%f 的位置上输出表达式 c 的值，最后的“\n”是 C 语言中特定的转义字符“回车换行”，使得屏幕光标换行，跳到下一行。

7.1.3　使用 printf 函数输出结果

（1）printf 的输出格式为自由格式，是否在两个数之间留逗号、空格或回车，完全取决于格式控制，如果不注意，很容易造成数字连在一起，使得输出结果没有意义。例如：若 m=10，n=12，则“printf("%d%d\n",m,n);”语句的输出结果是 1012，无法分辨 m 的值为 10，还是为 1012。而如果改为“printf("%d,%d\n",m,n);”其输出结果是“10,12”，看起来就一目了然了。

（2）格式控制中必须含有与输出项一一对应的输出格式说明，类型必须匹配。若格式说明与输出项的类型不一一对应匹配，则不能正确输出，而且编译时不会报错。若格式说明个数少于输出项个数，则多余的输出项不予输出；若格式说明个数多于输出项个数，则将输出一些毫无意义的数字乱码。

（3）如果要输出%符号，可以在格式控制中用%%表示，将输出一个%符号。

【例题 1】已知 i 是 123，f 是 34.5，写出下面的输出结果_____。

```
printf("%6d\n",i);
printf("%7.3f",f);
```

例题分析

第一个输出语句的结果是␣␣␣123（␣表示空格），第二个输出语句的结果是 34.500。

【例题 2】程序段“int x=12;double y=3.141593;printf("%d%8.6f",x,y);”的输出结果是（　　）。（2009 年 3 月）

A）123.141593　　B）12 3.141593　　C）12,3.141593　　D）123.1415930

例题分析

此题关键要看 printf 里面的内容，其中没有空格就不会中间输出空格，所以两个数应该靠在一起，而后面有空格，答案选 A。

【例题 3】有以下程序：（2009 年 9 月）

```
#include <stdio.h>
main()
{
   int a=1,b=0;
   printf("%d,",b=a+b);
   printf("%d\n",a=2*b);
}
```

程序运行后的输出结果是（　　）。

A）0,0　　B）1,0　　C）3,2　　D）1,2

例题分析

执行第一个输出语句得到 b 的值为 0+1=1，然后执行第二个输出语句 a=2*b，得到 2，选 D。

7.2　格式化输入 scanf()函数

在前面的程序中，看到的都是变量值的输出，而没有看到变量值的输入，计算机程序应该是交互的，也就是说可以让用户从键盘上输入，而此函数实现的就是这个功能。

7.2.1　基本格式

Scanf()函数称为格式输入函数，即按用户指定的格式从终端（比如键盘）上把数据输入到指定的变量，存放在计算

机内存中。scanf()函数的一般形式为：

scanf（“格式描述串”，地址表列）；

（1）格式描述串的作用与 printf()函数相同，但不能显示非格式字符串，也就是不能显示提示字符串。

（2）地址表列中给出各变量的地址，地址是由地址运算符&后跟变量名组成的。例如，&a，&b 分别表示变量 a 和变量 b 的地址，这个地址就是编译系统在内存中给 a，b 变量分配的地址。在 C 语言中，使用地址这个概念，这与其他语言不同。应该把变量的值和变量的地址这两个不同的概念区别开来。变量的地址是 C 编译系统分配的，用户不必关心具体的地址是多少。

7.2.2 格式说明

scanf()的格式描述串的一般形式为：

%[输入数据宽度][长短类型][长度]类型

其中有方括号[]的项为任选项。

类型：表示输入数据的类型，其格式符和意义如表 7-4 所示。

表 7-4 输入格式描述串中的类型

格式	字符意义
d	输入十进制整数
o	输入八进制整数
x	输入十六进制整数
u	输入无符号十进制整数
f 或 e	输入实型数（用小数形式或指数形式）
c	输入单个字符
s	输入字符串

说明：

（1）在格式串中，必须含有与输入项一一对应的格式转换说明符。若说明与输入项的类型不一一对应匹配，则不能正确输入，而且编译时不会报错。若格式说明个数少于输入项个数，scanf 函数结束输入，则多余的输入项将无法得到正确的输入值；若格式转换说明个数多于输入项个数，scanf 函数也结束输入，多余的数据作废，不会作为下一个输入语句的数据。

（2）在 VC 6.0 环境下，要输入 double 型数据，格式控制必须用%lf（或%le）。否则，数据不能正确输入。

（3）scanf 函数的格式字符前可以加入一个正整数指定输入数据所占的宽度，但不可以对实数指定小数位的宽度。

（4）若输入数据时采用如“%d%d”输入时，数据之间则可用空格或者回车做间隔。

（5）若输入数据时采用如“%d,%d”输入时，数据之间的输入间隔采用“,”分开。

7.2.3 通过 scanf 函数输入数据

1. 输入数值数据

在输入整数或实数这类数值型数据时，输入的数据之间必须用空格、回车符、制表符（Tab 键）等间隔符隔开，间隔符个数不限。即使在格式说明中人为指定了输入宽度，也可以用此方式输入。

例如，若 a 为 int 型变量，b 为 float 型变量，有以下输入语句：

```
scanf(“%d%f%le”,&a,&b,&c);
```

若要给 a 赋值 10，b 赋值 12.3，输入格式可以是

```
10␣12.3↙
```

也可以是：

```
10↙
12.3↙
```

2. 指定输入数据所占的宽度

可以在格式字符前加入一个正整数指定输入数据所占的宽度。例如上例可改为：

Scanf(“%3d%5f”,&a,&b);

若从键盘上输入：

123456.789.12356

用 printf(“%d　%f \n”,a,b,);打印的结果是：

123 456.700000

可以看到，由于格式控制是%3d，因此把输入数字串的前 3 位 123 赋值给了 a；由于对应于变量 b 的格式控制是%5f，因此把输入数字串中随后 5 位数（包括小数点）456.7 赋值给了 b。

由以上示例可知，数字之间不需要间隔符，若插入了间隔符，系统也将按指定的宽度来读取数据，从而会引起输入混乱。除非数字是“粘连”在一起，否则不提倡指定输入数据所占的宽度。

3. 在格式控制字符串中插入其他字符

scanf 函数中的格式控制字符串是为了输入数据用的，无论其中有什么字符，也不会输出到屏幕上，因此若想在屏幕上输出提示信息，应该首先使用 printf 函数输出。例如，

使用以下的形式。

```
int x,y,z;
printf("Please input x,y,z: ");
scanf("%d%d%d",&x,&y,&z);
```

运行时，由于 printf 语句的输出，屏幕上将出现提示“Please input x,y,z:”，只需按常规输入下面的数据即可。

12 34 56

【考生注意】

为了减少不必要的麻烦，尽量简化输入的方式，不要在输入语句中加入过多的通配符，以免出错。

【例题 4】如何给下面的程序输入 3 个数据（假设要输入的是 3 4 7）。

（1）scanf("%d%d%d",&a, &b, &c);

（2）scanf("%d,%d,%d",&a, &b, &c);

例题分析

（1）采用 3 4 7↙或者 3↙4↙7↙

（2）采用 3,4,7↙

7.3　单个字符的输入和输出 getchar()和 putchar()函数

7.3.1　单个字符输入函数

putchar()函数是字符输出函数，其功能是在显示器上输出单个字符。其一般形式为：

putchar（字符变量）;

【例题 5】

假设字符 c 的值为‘a’，输出字符 c 的值。

例题分析

直接使用 putchar(c);

7.3.2　单个字符输出函数

getchar()函数的功能是从键盘上输入一个字符。其一般形式为：

getchar();

通常把输入的字符赋予一个字符变量，构成赋值语句，如：

char c;

c=getchar();

【例题 6】从键盘输入一个大写字母，要求改用小写字母输出。

例题分析

小写字母与大写字母之间的 ASCII 码值差 32，也就是说相同的一个字母的小写比大写多 32。

```
#include <stdio.h>
void main()
{
   char c1,c2;
   c1=getchar();
   printf("%c,%d\n",c1,c1);
   c2=c1+32;
   printf("%c,%d\n",c2,c2);
}
```

7.4 习题

7.4.1 选择题

1. 设有定义“int a;float b;”，执行“scanf("%2d%f",&a,&b);”语句时，若从键盘输入 876543.0<回车>，a 和 b 的值分别是（　　）。

A）876 和 543.000000　　B）87 和 6.000000

C）87 和 543.000000　　D）76 和 543.000000

2. 以下程序的功能是：给 r 输入数据后计算半径为 r 的圆面积 s。程序在编译时出错。

```
#include <stdio.h>
main()
  /*Beginning*/
 {
    int r;  float s;
    scanf("%d",&r);
    s=π*r*r;
    printf("s=%f\n",s);
}
```

出错的原因是（　　）。

A）注释语句书写位置错误　　B）存放圆半径的变量 r 不应该定义为整型

C）输出语句中格式描述符非法　　D）计算圆面积的赋值语句中使用了非法变量

3. 若有说明语句“double *p,a;”，则能通过 scanf 语句正确给输入项读入数据的程序段是（　　）。

A）*p=&a; scanf("%1f",p);　　B）*p=&a; scanf("%f",p);

C）p=&a; scanf("%1f",*p);　　D）p=&a; scanf("%1f",p);

4. 有以下程序：

```
#include <stdio.h>
mian()
{
   int a=0,b=0;
   a=10;                          /*给 a 赋值
   b=2;                           给 b 赋值*/
   printf("a+b=%d\n",a+b);        /*输出计算结果*/
}
```

程序运行后的输出结果是（　　）。

A）a+b=10　　B）a+b=30　　C）30　　D）出错

5. 有以下程序，其中%u 表示按无符号整数输出程序运行后的输出结果，输出结果是（　　）。

```
#include <stdio.h>
main()
{
  unsigned int x=0xFFFF ;      /*x 的初值为十六进制数*/
  printf ("%u\n",x);}
```

A）-1　　B）65535　　C）32767　　D）0xFFFF

6. 执行以下程序段的输出结果是（　　）。

```
int m=0x12,n=0x12;
m=m-n;
printf("%X\n",m);
```

A）0x0　　B）0x12　　C）0x0　　D）0

7. 若 a，b，c，d 都是 int 型变量且都已经正确赋初值，则下列选项中不正确的赋值语句是（　　）。

A）a+b;　　B）a++;　　C）a=b=c=d=1000;　　D）a=(b=3)+(d=5);

8. 下列选项中合法的 C 语言赋值语句是（　　）。

A）a=b=1　　B）a=int a+b;　　C）a=2,b=3　　D）i++;

9. 下列选项中合法的赋值语句是（　　）。

A）a=b=34　　B）a=34,b=34　　C）i-1;　　D）m=(int) (x+y);

10. 若 k 为 int 型变量，以下程序段执行后的输出如果是（　　）。

```
#include 〈stdio.h〉
main()
{
   int k=3;
   if(k)
     printf("###");
   else
     printf("&&&&");
}
```

A）###　　B）&&&&　　C）###&&&&　　D）有语法错误，不能执行

11. 若以下程序

```
#include 〈stdio.h〉
main()
{  char str[10];
   scanf("%s",&str);
   printf ("%s\n",str);
}
```

运行上面的程序，输入字符串“how are you”，则程序的执行结果是（　　）。

A）how　　B）how are you　　C）h　　D）howareyou

12. 下列写法中正确的是（　　）。

A） main ()　　B）main()　　C）main()　　D）main()

{ int i =3,j}　　{int i=3;　　{;}

13. 变量已正确定义，下面程序段的输出结果是（　　）。

```
#include 〈stdio.h〉
main()
{
   float x=1.236547;
   printf( " %f\n",( int ) ( x*1000+0.5)/(float)1000);
}
```

A）1.237000

B）输出格式说明与输出项不匹配，输出无定值

C）1.236000

D）1.24

14. 设已定义 x 为 double 类型变量，若以下程序：

```
#include 〈stdio.h〉
main()
{
   float x=323.82631;
   printf("%.2e\n",x) ;
}
```

则对以上程序段（　　）。

A）输出格式描述符的域宽不够，不能输出

B）输出为 32.38e+01

C）输出为 3.24e+002

D）输出为 3.24e2

15. 设有定义“long x =123450L;”，则下列输出语句中不能够正确输出变量 x 的是（　　）。

A）printf("x= %d\n" ,x);　　B）printf("x= %id\n" ,x);

C）printf("x= %dL\n" ,x);　　D）printf("x= %1d\n" ,x);

16. 若有以下定义和语句：

```
int u=011,v=0x11,w=11;
printf("%o,%x,%d\n", u,v,w);
```

则输出结果为（　　）。

A）9,17,11　　B）9,11,11　　C）11,11,11　　D）11,17,11

17. 设已定义 a 为 int 类型变量：

```
#include <stdio.h>
#include <string.h>
main()
{ int a=-1234;
  printf( "5222%d\n" ,a);
}
```

则以上语句（　　）。

A）输出为 5222-1234　　B）输出为-1234

C）格式描述符不合法，输出无定值　　D）输出为 1234

18. 若有定义“int a=3;”和输出语句“printf("%8x",a);”，则以下正确的叙述是（　　）。

A）整型变量的输出格式符只有%d 一种

B）%x 是格式符的一种，它可以适用于任何一种类型的数据

C）%x 是格式符的一种，其变量的值按十六进制数输出，但%8x 是错误的

D）%8x 是正确的格式符，其中数字 8 规定了输出字段的宽度

19. 若变量已正确定义，要求通过“scanf("%c%d%c%d",&c1,&a,&c2,&b);”语句给变量 a 和 b 赋值 32 和 45，变量 c1 和 c2 赋字符‘A’和‘B’，则以下选项中数据从第 1 列开始输入，正确的输入形式是（　　）。

A）A32 ＜回车＞　　B）A45＜回车＞

B45＜回车＞　　B32＜回车＞

C）A32B45＜回车＞　　D）A 32 B 45＜回车＞

20. 若有“double a;”，则正确的输入语句是（　　）。

A）scanf("%1f",a);　　B）scanf("%f",&a);　　C）scanf("%1f",&a);　　D）scanf("%1e",&a);

21. 已知 i，j，k 为 int 型变量，若要从键盘输入 2，3，4＜回车＞，使 i，j，k 的值分别为 2，3，4，以下正确的输入语句是（　　）。

A）scanf("%3d,%3d,%3d",&i,&j,&k);

B）scanf("%d,% d,% d",&i,&j,&k);

C）scanf("%d%d%d",&i,&j,&k);

D）scanf("i=%d,j=%d,k=%d",&i,&j,&k);

22. 在 scanf 函数的格式控制中，格式说明的类型与输入项的类型应该对应匹配。如果类型不匹配，系统将（　　）。

A）不予接收

B）不能得到正确的数据，且不给出出错信息

C）能接收到正确输入

D）给出出错信息，不予接收输入

7.4.2　填空题

1．以下程序运行时若从键盘输入：10 20 30 后，按 Enter 键。则输出结果为_____。

```
#include <stdio.h>
main()
{   int   i=0,j=0,k=0;
```

```
    scanf("%d%*d%d",&i,&j,&k); printf ("%d%d%d\n" ,i,j,k);
}
```

2. 以下程序运行后的输出结果为_____。

```
#include <stdio.h>
main()
{ int  x=0210; printf("%x\n",x);
}
```

3. 设有定义“float x=123.4567;”，则执行以下语句后的输出结果为_____。

```
printf("%f\n",(int)(x*100+0.5)/100.0);
```

4. 以下程序运行后的输出结果是_____。

```
#include <stdio.h>
main()
{ int m=011,n=11;
  printf("%d  %d\n",++m,n++);
}
```

5. 以下程序运行后的输出结果是_____。

```
#include <stdio.h>
main()
{
   char c; int n=100;
   float f=10;double x;
   x=f*=n/=(c=50);
   printf("%d  %f\n",n,x);
}
```

第8章 选择结构程序设计

本章主要介绍关系运算符及其表达式，逻辑运算符及其表达式，以及 if 语句和 switch 语句，结合计算机等级考试要求，具体如表 8-1 所示。

表 8-1　考试要求

考试知识点	重要性
关系运算符与关系表达式	★
逻辑运算符与逻辑表达式	★★
if 语句和条件运算	★★★★
switch 语句，break 语句	★★★

8.1　关系运算符及其表达式

通俗地说，关系运算符实际就是连接两个数据进行比较的符号，而关系表达式就是利用关系运算符连接比较数据的组合式。

8.1.1　关系运算符及其优先次序

C 语言提供了 6 种关系运算符，如表 8-2 所示。

表 8-2　关系运算符

<	<=	>	>=	= =	!=
小于	小于等于	大于	大于等于	等于	不等于

其中，前 4 种关系运算符（<，<=，>=，>）的优先级相同，后两种也相同，并且，前 4 种的优先级高于后面的两种。另外，关系运算符的优先级要低于算术运算符，而高于赋值运算符。

【考生注意】在 C 语言中= =表示的是关系运算符中的等于，而=表示赋值。

8.1.2　关系表达式

1. 关系表达式的概念

所谓关系表达式是指将两个表达式（可以是算术表达式、关系表达式、逻辑表达式、赋值表达式或字符表达式）连

接起来，进行关系运算的式子。这里所说的逻辑表达式在后面的 8.2 节就会讲到。

2. 关系表达式的值

每一个关系表达式都是有值的，这个值是一个逻辑值（真或假），由于 C 语言没有逻辑型数据，所以用整数 1 表示逻辑真，用整数 0 表示逻辑假。

关系运算符举例如下。

```
#include <stdio.h>
void main()
{
  int num1=1,num2=3,num3=4;
  printf("%d",num1>num2);  /*输出为 0*/
  printf("%d",num1>num2-2); /*输出为 0*/
  printf("%d",num1>(num3-3)); /* 输出为 0*/
  printf("%d",num1== num2>num3);/* 输出为 0*/
  printf("%d", num1=(num2<num3+2));/* 输出为 1*/
}
```

8.2 逻辑运算符及其表达式

逻辑运算符在 C 语言中占据重要的地位，当程序中出现两个以上条件的时候，那就要用到逻辑运算的内容了。

8.2.1 逻辑运算符及优先次序

1. 逻辑运算符

C 语言提供了 3 种逻辑运算符，如表 8-3 所示。

表 8-3 逻辑运算符

逻辑与	逻辑或	逻辑非
&&	||	!

其中，&&和||是双目运算符，也就是要有两个表达式，而！是单目运算符，只要有一个表达式。

2. 运算符的运算规则

（1）&&：当且仅当两个表达式的值都是逻辑真（为 1）的时候，运算结果为逻辑真（为 1），否则为逻辑假（为 0）。

（2）||：当且仅当两个表达式的值都是逻辑假（为 0）的时候，运算结果为逻辑假（为 0），否则为逻辑真（为 1）。

（3）!：当一个表达式的值为逻辑真（为非 0）的时候，运算结果为逻辑假（为 0），否则就为逻辑真（为 1）。

说明：

在数学上关系式 0<x<5 表示 x 的值在大于 0 并且小于 5 的范围内，但在 C 语言中不能直接用 0<x<5 这样一个关系表达式来表述以上的逻辑关系。对于这种情况，只有采用 C 语言提供的逻辑表达式 0<x&&x<5 才能正确表述以上关系。

8.2.2 逻辑表达式

1. 逻辑表达式的概念

所谓逻辑表达式是指用逻辑运算符将一个或多个表达式连接起来，进行逻辑运算的式子。

2. 逻辑表达式的值

逻辑运算表达式的结果如表 8-4 所示。

表 8-4　逻辑运算表达式结果

表达式 A	表达式 B	A&&B	A‖B	!A	!B
逻辑真（1）	逻辑真（1）	逻辑真（1）	逻辑真（1）	逻辑假（0）	逻辑假（0）
逻辑真（1）	逻辑假（0）	逻辑假（0）	逻辑真（1）	逻辑假（0）	逻辑真（1）
逻辑假（0）	逻辑真（1）	逻辑假（0）	逻辑真（1）	逻辑真（1）	逻辑假（0）
逻辑假（0）	逻辑假（0）	逻辑假（0）	逻辑假（0）	逻辑真（1）	逻辑真（1）

【例题 1】逻辑运算符举例。

```
#include <stdio.h>
void main()
{
   int a=3,b=2,c=4,m=6,t;
   printf("%d",a<b&&b<c);    /*输出为 0*/
   printf("%d",a<b||b>c);    /*输出为 0*/
   printf("%d,%d", (a>b)&&(m=c),m); /*输出为 1, 4*/
   printf("%d,%d", (a<b)&&(m=c),m); /*输出为 0, 6*/
   printf("%d,%d", (a<c)||(m=c),m); /*输出为 1, 6*/
}
```

【考生注意】

（1）在使用&&运算符的时候，如果&&运算符左边的逻辑表达式的结果为 0 的时候，那么&&运算符右边的表达式无须计算，因为无论&&运算符的右边表达式的结果是 1 还是 0，结果都是 0（如上面的第 3 个输出语句，第 4 个输出语句）。

（2）在使用‖运算符的时候，如果‖运算符左边的逻辑表达式的结果为 1，那么此逻辑运算符右边的表达式不需要计算，因为无论右边的逻辑表达式的结果为 1 还是为 0，其结果都是 1（如上面的第 5 个输出语句）。

（3）在逻辑运算中出现赋值运算表达式的时候，如果经过赋值后该赋值变量为非 0 的数，那么赋值表达式的值为逻辑真，否则为逻辑假。

【例题 2】若变量已经被正确定义，为表示变量 x 和 y 都能被 3 整除，应该使用的表达式是（　　）。

A）（x%3！=0）‖（y%3！=0）

B）（x%3！=0）&&（y%3！=0）

C）（x%3==0）‖（y%3==0）

D）（x%3==0）&&（y%3==0）

例题分析

首先从题目的意思上理解，所谓整除，指的是余数为 0，变量 x 和 y 都能被 3 整除，所以 x%3= =0 和 y%3= =0，由于都能被整除，所以必须用&&进行连接。答案选 D。

8.3　if 语句和条件运算

if 语句是用来判定所给定的条件是否满足，而执行不同的语句。

8.3.1 if 语句

1. if 语句的格式

if（条件表达式）

{

语句 1;

语句 2;

……

语句 n;

}

说明：

（1）if 语句中的条件表达式主要是由前面所讲的关系表达式、逻辑表达式、赋值表达式或者是其他类型的数据等组成，并且条件表达式是由一对()括起来的。

（2）一对{}可以加也可以不加，加上这一对大括号的意思表示在条件表达式为真的时候，需要执行的语句在两条或两条以上，不加大括号的意思表示在条件表达式为真的时候，需要执行的语句只有一条。

2. 执行的过程

if 语句执行过程如图 8-1 所示。

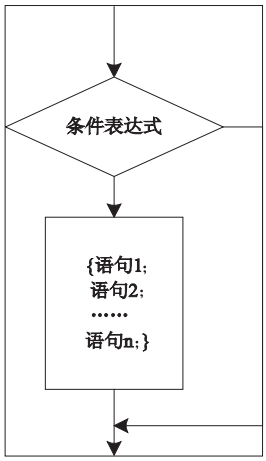


图 8-1 if 语句执行过程

3. 在使用 if 语句时还应注意以下问题

（1）在 3 种形式的 if 语句中，在 if 关键字之后均为表达式。该表达式通常是逻辑表达式或关系表达式，但也可以是其他表达式，如赋值表达式等，甚至也可以是一个变量。例如 if(a=5) 语句和 if(b)语句都是允许的。只要表达式的值为非 0，即为真。如在“if(a=5)…;”中表达式的值永远为非 0，所以其后的语句总是要执行的，当然这种情况在程序中不一定会出现，但在语法上是合法的。

（2）在 if 语句中，条件判断表达式必须用括号括起来，在语句之后必须加分号。

（3）在 if 语句的 3 种形式中，所有的语句应为单个语句，如果要想在满足条件时执行一组（多个）语句，则必须把这一组语句用{}括起来组成一个复合语句。但要注意的是在}之后不能再加分号。

【例题 3】以下是 if 语句的基本形式：if（表达式）语句其中“表达式”（　　）。（2009 年 3 月）

A）必须是逻辑表达式　　B）必须是关系表达式

C）必须是逻辑表达式或关系表达式　　D）可以是任意合法的表达式

例题分析

表达式只要是合法的表达式都可以。答案选 D。

【例题 4】

```
#include <stdio.h>
 void main（）
{
   int m=1,n=2,i;
   scanf("%d",&i);
   if（i>2）
{
   m=m*2;
```

```
    n=n+1;
  }
    m=m+n;
    printf("%d",m);
  }
```

（1）当输入 1 的时候，结果是多少？

（2）当输入 4 的时候，结果是多少？

例题分析

当输入 1 的时候，由于 i 的值为 1，i>2 的值为逻辑假，所以条件表达式不成立，所以大括号中的语句就不执行，而去执行大括号后面的语句，即 m＝m＋n，结果为 3。当输入 4 的时候，由于 i 的值为 4，4>2 的值为逻辑真，所以条件表达式成立，执行大括号中的语句（m 为 2，n 为 3），然后再去执行 m＝m＋n，结果为 5。

【例题 5】有以下程序：（2009 年 3 月）

```
#include <stdio.h>
void main（）
{ int  x;
  scanf("%d",&x);
  if(x<=3);
  else if(x!=10) printf("%d\n",x);
}
```

程序运行时，输入的值在哪个范围才会有输出结果？

A）不等于 10 的整数　　B）大于 3 且不等于 10 的整数

C）大于 3 或等于 10 的整数　　D）小于 3 的整数

例题分析

此题主要是考查对 if else 的理解，在本题中，容易忽略的是第 1 个 if 后面有分号，这是一个空语句，而输出语句被 else if 中的 if 控制着，所以要想输出结果，必须执行 if(x!=10)，同时，不执行 if(x<=3)，所以选 B。

【例题 6】以下程序段中，与语句“k=a>b?(b>c?1:0):0;”功能相同的是_____。（2009 年 9 月）

A）if((a>b)&&(b>c)) k=1;
　　else k=0;

B）if((a>b)||(b>c) k=1;
　　else k=0;

C）if(a<=b) k=0;
　　else if(b<=c) k=1;

D）if(a>b) k=1;
　　else if(b>c) k=1;
　　else k=0;

例题分析

题目的意思是当 b 大于 c 的时候，结果为 1，而 a 大于 b 的时候，结果也是 1，所以本题所表示的意思是 a 大于 b，b 大于 c，所以选 D。

【例题 7】输入三角形的三边长，求三角形面积。

例题分析

三边长为 a，b，c，面积 area=s(s-a)(s-b)(s-c)，其中 s=(a+b+c)/2。

```
#include <math.h>
#include  "stdio.h"
void main()
{
  float a,b,c,s,area;
  sacnf ("%f,%f,%f",&a,&b,&c);
  s=1.0/2*(a+b+c);
  area=sqrt(s*(s-a)*(s-b)*(s-c));
  printf("a=%7.2f,b=%7.2f,c=%7.2f,s=%7.2f\n",a,b,c,s);
  printf("area=%7.2f\n",area);}
```

【例题 8】例如：从键盘上输入字符，判别其分类：控制符、数字、大写字母、小写字母、其他字符。

```
#include "stdio.h"
void main()
{
   char c;
   c=getchar();
   if(c<32)
      printf("This is a control character.\n");
   else if(c>='0'&&c<='9')
      printf("This is a digit.\n");
   else if(c>='A'&&c<='Z')
      printf("This is a capitalletter.\n");
   else if(c>='a'&&c<='z')
      printf("This is a small letter.\n");
   else
      printf("This is an other charater.\n");
}
```

【例题 9】编写程序，要求从键盘上输入一个 x，并计算下列 y 的值（保留 2 位小数）。

$$y=\begin{cases} x+3 & x<-2 \\ 4-\sin(x) & -2\leqslant x\leqslant 1 \\ x*x+1 & x>1 \end{cases}$$

例题分析

题目中 x 有多种范围，并且对应 y 有不同的值，所以当我们输入 x 后，根据 x 的值就可以进行条件判断，从而选择适合 x 条件的语句，然后就能求出对应的 y 值，这显然是一个条件判断的结构。所以，在这里我们选择 if else 语句。当 x<-2 的时候，我们执行 x+3，当 x≥-2 的时候，出现了两种情况，一种是在 x≤1 的时候，执行 4-sin(x)，另一种是在 x>1 的时候，执行 x*x，所以很显然在 x≥-2 的时候要涉及一个 if else 的嵌套。同时，要注意使用头文件 math.h，因为这里有 cos(x)函数。

【例题代码】

```
#include  "math.h"
#include  "stdio.h"
void main()
{
```

```
    float x,y;
    scanf("%f",&x);
    if(x<-2)
      y=x+3;
  else if（x<=1）/*在这里只要写上 x<=1 就可以了，不需要写成 x>=-2&& x<=1，因为这个 if 语句就属于 else 的范围了，而 else 表示的意思就是 x≥-2*/
        y=4-sin(x);
      else
        y=x*x+1;
    printf("y=%.2f",y);/*%.2f 表示输出的数有两位小数*/
  }
```

8.3.2　if else 语句

1. if else 语句的格式

if（条件表达式）

{

语句 1；

语句 2；

……；

语句 n；

}

else

{

语句 1；

语句 2；

……；

语句 n；

}

说明：

（1）这里的 if 条件表达式与前面所讲的 if 条件表达式完全一样。

（2）else 所表示的条件指的是 if 条件表达式中的反方向。

（3）else 不能脱离 if 而单独存在。

2. 执行过程

执行过程如图 8-2 所示。

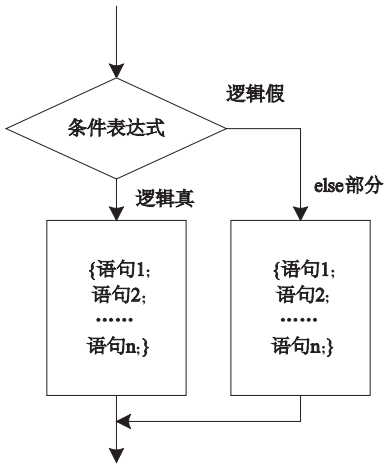


图 8-2　if else 语句执行过程

8.3.3　if 语句的嵌套

所谓 if 语句的嵌套是指在 if 语句中又包含一个或多个 if 语句。

if 语句嵌套的一般格式

```
if（条件表达式）
    if（条件表达式）
        {语句 1；
         ……
         语句 n；
        }
    else
        {语句 1；
         ……
         语句 n；
        }
 else
    if（条件表达式）
        {语句 1；
         ……
         语句 n；
        }
    else
        {语句 1；
         ……
         语句 n；
        }
```

【考生注意】

if else 的配对关系，从最内层开始，else 总是与它上面最近的（未配对）的 if 配对，请注意其中的直线，表示 else 与 if 的配对关系。

【例题 10】写出下面程序的结果。

```
#include  "stdio.h"
void main()
{
   int a=2,b=3,c=1;
   if(a>1)
   if(c>2)
      c=c+1;
   else
      c=b-c;
   printf("a=%d,b=%d,c=%d",a,b,c);
}
```

例题分析

此题主要考查对嵌套的 if else 语句的理解，从题目上看，首先执行 if(a>1)，由于 a>1，所以进一步执行 if(c>2)，由于 c<2，所以 c=c+1 不执行，而是去执行 else 部分的 c=b-c，得到 c 的值为 2，最后 a，b，c 的值是 2，3，2。

【考生注意】

在【例题 10】中，很多考生会误认为 if(a>1)是和 else 相对应的，其实和 else 相对应的是 if(c>2)，想想为什么？

【例题 11】输入 3 个整数 x，y，z，请把这 3 个数由小到大输出。

例题分析

对于 3 个数的比较，先将 x 与 y 进行比较，如果 x>y 则将 x 与 y 的值交换，然后再用 x 与 z 进行比较，如果 x>z 则将 x 与 z 的值交换，这样能使 x 最小。

```
#include  "stdio.h"
void main()
{
   int x,y,z,t;
   scanf("%d%d%d",&x,&y,&z);
   if(x>y)
     {t=x;x=y;y=t;} /*交换 x，y 的值*/
   if(x>z)
     {t=z;z=x;x=t;}/*交换 x，z 的值*/
   if(y>z)
     {t=y;y=z;z=t;}/*交换 z，y 的值*/
   printf("small to big: %d %d %d\n",x,y,z);
}
```

【例题 12】写一个程序，对输入的小写字母循环后移 5 个位置，如‘a’变成‘f’，‘w’变成‘b’。

例题分析

本程序主要是将字母往后移动 5 个位置，也就是 a 往后移动 5 个位置变成 f，把 b 往后移动 5 个位置变成 g，一直这样下去，如图 8-3 所示。

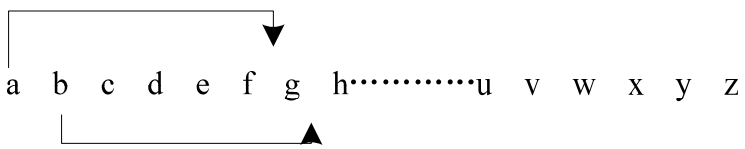


图 8-3　字符 a 到 u 的移动过程

如果求 a 往后移动 5 个位置，只需要把 a+5 就可以了，但是如果一直到 v 该怎么办呢？把 v 往后移动 5 个位置，就会移动到 z 后面的字符，那么如何得到 a 呢？其实很简单，我们只需要把 v+5-26 就可以了，这是因为 a 到 z 一共有 26 个字母，为了能得到 a，所以把 v+5 往前移动 26，这样就可以指向 a 了。如图 8-4 所示。

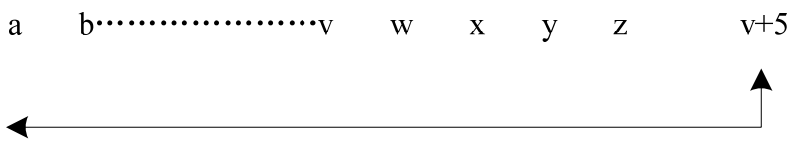


图 8-4　字母 v 到 z 的移动过程

从上面的这个示意图中我们知道 v 到 z 都必须注意这个问题，所以这几个字母我们要进行特殊处理。

【例题代码】

```
#include  "stdio.h"
```

```
void main()
{
   char c;
   c=getchar();    /*输入一个字母*/
   if(c>='a'&&c<='u')
     c=c+5;
   else if(c>='v'&&c<='z')
          c=c+5-26;
   putchar(c);
}
```

说明：

在本题中需要注意的地方是 if else 语句的使用，我们经常可能会写成如下形式：

```
if(c>='a'&&c<='u')
   c=c+5;
if(c>='v'&&c<='z')
   c=c+5-26;
```

看起来好像没有错，但如果仔细检查就会发现上面的错误，假设有一个变量 c 为 r，按照上面的语句执行就会变成 w，其实到这里按照题目本意已经执行完了，但是从程序中可以看出还必须往下执行下面的 if 语句，因为 w 正好符合下面这个 if 条件，所以就会将 w 变成 b 了，所以输入一个 r，本来该得到 w，反而得到了 b。这是我们需要注意的地方。

【例题 13】企业发放的奖金根据利润提成。利润（I）低于或等于 10 万元时，奖金可提 10%；利润高于 10 万元，低于 20 万元时，低于 10 万元的部分按 10%提成，高于 10 万元的部分，可提成 7.5%；20 万到 40 万之间时，高于 20 万元的部分，可提成 5%；40 万元到 60 万元之间时高于 40 万元的部分，可提成 3%；60 万元到 100 万元之间时，高于 60 万元的部分，可提成 1.5%；高于 100 万元时，超过 100 万元的部分按 1%提成。从键盘输入当月利润 I，求应发放奖金总数？

例题分析

本题要使用多个 if else 语句进行表达。

```
#include "stdio.h"
main()
{
   long int i;
   int bonus1,bonus2,bonus4,bonus6,bonus10,bonus;
   scanf("%ld",&i);
   bonus1=100000*0.1;bonus2=bonus1+100000*0.75;
   bonus4=bonus2+200000*0.5;
   bonus6=bonus4+200000*0.3;
   bonus10=bonus6+400000*0.15;
     if(i<=100000)
       bonus=i*0.1;
     else if(i<=200000)
            bonus=bonus1+(i-100000)*0.075;
          else if(i<=400000)
                 bonus=bonus2+(i-200000)*0.05;
               else if(i<=600000)
```

```
                    bonus=bonus4+(i-400000)*0.03;
            else if(i<=1000000)
                        bonus=bonus6+(i-600000)*0.015;
                 else
                        bonus=bonus10+(i-1000000)*0.01;
   printf("bonus=%d",bonus);
}
```

8.3.4　条件表达式

条件表达式也是一种具有选择结构的表达式，与前面所讲的 if else 语句所表示的选择是同一个意思。

表达式=表达式 1？表达式 2：表达式 3；

说明：

（1）当表达式 1 的值为逻辑真，则整个条件表达式的值为表达式 2。

（2）当表达式 1 的值为逻辑假，则整个条件表达式的值为表达式 3。

使用条件表达式时，还应注意以下几点。

（1）条件运算符的运算优先级低于关系运算符和算术运算符，但高于赋值运算符。

（2）条件运算符?和:是一对运算符，不能分开单独使用。

（3）条件运算符的结合方向是自右至左。

【例题 14】从键盘上输入 4，写出下列程序的结果。

```
#include  "stdio.h"
void main()
{
   int n,y;
   scanf("%d",&n);
   y=n>3？n+3:n-2;
   printf("%d",y);
}
```

例题分析

此题是一个关于条件表达式的考题，很显然 4>3，所以 y=n+3，即 7，所以 y 的值是 7。

8.4　switch 语句

利用 if 语句的基本形式，可以实现只有两个分支的选择，利用 if 语句的嵌套形式，可以实现多分支的选择。但分支越多，嵌套的层数就越多，导致程序变得很长，因此 C 语言提供了 switch 语句来直接处理多分支选择。

1. switch 语句的格式

switch（表达式）

{

case 常量表达式 1：语句 1；[break]；

case 常量表达式 2：语句 2；[break]；

……

case 常量表达式 n：语句 n；[break]；

```
    default：          语句 n＋1；
}
```

说明：

（1）switch 后面的表达式可以是整型表达式或字符型表达式。

（2）当 switch 后面的表达式的值与 case 中的某个常量表达式的值相等的时候，就会去执行其后面的语句。

（3）break 语句由一对中括号括起表示可写可不写。break 表示退出 switch 语句。当没有 break 语句的时候，程序会一直执行下一条语句，直到执行到有 break 语句或全部语句结束。

（4）default 表示当 switch 后面的表达式不等于 case 中的常量表达式的值的时候，执行语句 n＋1。

2. 使用 switch 语句时应注意的事项

（1）在 case 后的各常量表达式的值不能相同，否则会出现错误。

（2）在 case 后，允许有多条语句，可以不用{}括起来。

（3）各 case 和 default 子句的先后顺序可以变动，而不会影响程序执行结果。

（4）default 子句可以省略不用。

【例题 15】写出下面程序的结果。

```
#include "stdio.h"
void main()
{
   int n;
   scanf("%d",&n);
   switch(n)
    {
      case 1:n=n+1;break;
      case 2:n=n+2;
      default:n=n+3;
      case 3:n=n+4;
   }
   printf("n=%d",n);
}
```

（1）当输入 n 的值为 1 的时候，结果是多少？

（2）当输入 n 的值为 4 的时候，结果是多少？

例题分析

当输入 1 的时候，执行 csae 1，结果是 2，当输入 4 的时候，执行 default: n=n+3 和 case 3: n=n+4，得到结果是 10

【考生注意】

此题中 default 语句并不位于 switch 语句的最下面，有些考生就不知道该如何处理了，其实 default 语句的含义就是 case 中的其他部分。

【例题 16】计算器程序。用户输入运算数和四则运算符，输出计算结果。

```
#include "stdio.h"
void main()
{
   float a,b,s;
   char c;
```

```
    printf("input expression: a+(-,*,/)b \n");
    scanf("%f%c%f",&a,&c,&b);
    switch(c){
      case '+': printf("%f\n",a+b);break;
      case '-': printf("%f\n",a-b);break;
      case '*': printf("%f\n",a*b);break;
      case '/': printf("%f\n",a/b);break;
      default: printf("input error\n");
    }
}
```

8.5　习题

8.5.1　选择题

1. C 语言对嵌套 if 语句的规定是：else 总是与（　　）配对。

A）其之前最近的 if　　B）第一个 if

C）缩进位置相同的 if　　D）其之前最近且不带 else 的 if

2. 以下不正确的 if 语句形式是（　　）。

A）if (x>y && x!=y);

B）if(x==y) x+=y;

C）if(x!=y) scanf(“%d”,&x);
　else scanf(“%d”,&y);

D）if(x<y){x++;y++;}

3. 以下关于 switch 语句和 break 语句的描述中，只有（　　）是正确的。

A）在 switch 语句中必须使用 break 语句

B）break 语句只能用于 switch 语句

C）在 switch 语句中，可以根据需要使用或不使用 break 语句

D）break 语句是 switch 语句的一部分

4. 有如下程序：

```
#include <stdio.h>
main()
{ int x=1,a=0,b=0;
     switch(x)
     {
       case 0:b++;
       case 1:a++;
       case 2:a++;b++;
     }
     printf("a=%d,b=%d\n",a,b);
}
```

该程序的输出结果是（　　）。

A）a=2, b=1　　B）a=1, b=1　　C）a=1, b=0　　D）a=2, b=2

5. 下列程序的输出结果是（　　）。

```
#include <stdio.h>
main()
{ int a=0,i;
   for(i=1;i<5;i++)
   {  switch(i)
     { case 0:
       case 3:a+=1;
       case 1:
       case 2:a+=2;
       default:a+=3;
     }
  }
  printf("%d",a);
}
```

A）19　　B）1　　C）6　　D）8

6. 执行下列程序，输入为 1 时的输出结果是（　①　），输入为 3 时的输出结果是（　②　）。

```
#include <stdio.h>
main()
{   int k;
   scanf("%d",&k);
   switch(k)
   {
      case 1:printf("%d\n",k++);
      case 2:printf("%d\n",k++);
      case 3:printf("%d\n",k++);
      case 4:printf("%d\n",k++);
          break;
      default:printf("Full!\n");
   }
}
```

①A）1　　B）2　　C）2　　D）1

3　　2

4　　3

5　　4

②A）3　　B）4　　C）3　　D）4

4　　5

7. 有下列程序：

```
#include <stdio.h>
main()
```

```
{    int i,s=0,t[]={1,2,3,4,5,6,7,8,9};
     for(i=0li<9;i+=2)s+=*(t+i);
     printf("%d\n",s);
}
```

程序执行后的输出结果是（　　）。

A）45　　B）20　　C）25　　D）36

8.5.2　填空题

1. 以下程序在输入 5，2 之后的输出是_____。

```
#include <stdio.h>
main()
{  int s,t,a,b;
   scanf("%d,%d",&a,&b);
   s=1;
   t=1;
   if (a>0) s=s+1;         /* ① */
   if(a>b)  t=s+t;       /* ② */
   else if(a==b) t=5;
        else t=2*s;
   printf("s=%d,t=%d\n",s,t);
 }
```

2. 执行以下程序，输入-10 的结果是_____，输入 5 的结果是_____，输入 10 的结果是_____，输入 30 的结果是_____。

```
#include <stdio.h>
main()
{ int x,c,m;
  float y;
  scanf("%d",&x);
  if(x<0) c=-1;
  else c=x/10;
  switch(c);
  {  case -1:y=0;break;
     case 0:y=x;break;
     case 1:y=10;break;
     case 2:
     case 3:y=-0.5*x+20;break;
     default:y=-2;
  }
  if(y!=-2)printf("y=%g\n",y);
  else printf("error\n");
 }
```

3. 以下程序的执行结果是_____。

```
#include <stdio.h>
main()
{ int a=2,b=7,c=5;
  switch(a>0)
  {  case 1:switch(b<0)
         {  case 1:printf("@");break;
            case 2:printf("!");break;
         }
     case 0:switch(c==5)
         { case 0:printf("*");break;
           case 1:printf("#");break;
           default:printf("$");break;
         }
      default:printf("&");
     }
     printf("\n");
   }
```

4. 当 a=1，b=2，c=3 时，以下语句执行后，a，b，c 中的值分别为_____，_____，_____。

```
if(a>c)
b=a;a=c;c=b;
```

8.5.3 程序设计题

1. 编写程序判断输入的正整数是否既是 5 又是 7 的倍数，如果是，则输出 yes，否则输出 no。
2. 编写程序，从键盘上输入 4 个数，求出其中的最大值。
3. 输入 x，计算并输出下列分段函数的值，可以调用数学库中的绝对值函数 fabs()。

$$y=\begin{cases} |x| & x<-1 \\ 2-\sin(x) & -1=<x<=1 \\ \cos(x)+3 & 1<=x<=3 \\ x*x & x>3 \end{cases}$$

第9章 循环结构程序设计

本章主要介绍循环的概述、while 循环、do while 循环、for 循环，以及 break 语句和 continue 语句，结合计算机等级考试二级 C 语言要求，具体如表 9-1 所示。

表 9-1　考试要求

考试知识点	重要性
循环语句概述	★
while 循环	★★★
do while 循环	★★
for 循环	★★★★
break 语句，continue 语句	★★★

9.1 循环语句概述

在不少实际问题中有许多具有规律性的重复操作，因此在程序中就需要重复执行某些语句。一组被重复执行的语句称之为循环体，能否继续重复，取决于循环的终止条件。循环语句是由循环体和循环的终止条件两部分组成的。

9.2 for 语句和其构成的循环结构

1. 一般格式

for（[循环变量赋初值];[循环继续条件];[循环变量增值]）

　　{循环体语句组;}

如果循环体语句组仅由一条语句构成，可以不用复合语句形式。

2. 执行流程

执行流程如图 9-1 所示。

（1）求解“循环变量赋初值”表达式。

（2）求解“循环继续条件”表达式。如果其值非 0，转（3）；否则，转（4）。

（3）执行循环体语句组，并求解“循环变量增值”表达式，然后转（2）。

（4）执行 for 语句的下一条语句。

从下面的执行流程可以看出：“循环变量赋初值”表达式只求解 1 次，而“循环继续条件”、“循环变量增值”和循环体语句组，则要执行若干次（具体次数由“循环继续条件”表达式决定）。

说明：

（1）“循环变量赋初值”、“循环继续条件”和“循环变量增值”部分均可缺省，甚至全部缺省，但其间的分号不能省略。这 3 部分的功能即构成循环的 3 个基本条件，不在 for 语句行实现，就必须在适当的位置，以适当的方式来实现。

（2）“循环变量赋初值”表达式，既可以是给循环变量赋初值的赋值表达式，也可以是与此无关的其他表达式（如逗号表达式）。

（3）“循环变量增值”不一定是指增加，也可以指减少。

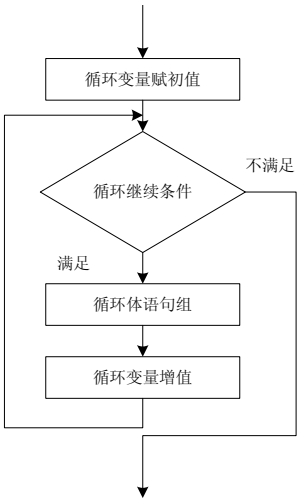


图 9-1　for 循环结构流程

【例题 1】编写程序，求出 1~100 之间所有 3 的倍数之和。

例题分析

在本题中，表示 3 的倍数就是某数除以 3 的余数是 0，显然，这个求和公式是重复执行的，所以采用循环来表示。

```
#include  "stdio.h"
void main()
{ int i,sum=0;
  for(i=1;i<=100;i++)
    if(i%3==0)
        sum=sum+i;
  printf("sum=%d",sum);
}
```

【例题 2】有以下程序：（2009 年 9 月)

```
#include <stdio.h>
void main()
{ int c=0,k;
  for(k=1;k<3;k++)
  switch(k)
  { default:c+=k;
    case 2:c++;break;
    case 4:c+=2;break;
  }
  printf("%d\n",c);
}
```

程序运行后的输出结果是（　　）。

A）3　　B）5　　C）7　　D）9

例题分析

本题考查将 for 语句和 switch 结合起来的知识点，首先执行 k=1 的时候，执行 switch 语句，执行 c+=k，得到 c=1，继续执行 case 2，执行 c++，得到 c=2，执行 break 语句，跳出 switch 语句，但此时还处于 for 循环的控制中，执行 k++，得到 k 的值为 2，再次执行 switch ()语句，执行 case 2，c++，得到 c 的值为 3，然后执行 break 语句，跳出 switch 语句，执行 k++，得到 k 的值为 3，不满足 for 循环条件，执行输出语句，输出 c 的值，得到答案为 A。

【例题 3】有以下程序：（2009 年 9 月)

```
#include  <stdio.h>
main()
```

```
{ int a[ ]={2,3,5,4},i;
  for(i=0;i<4;i++)
   switch(i%2)
  { case 0:switch(a[i]%2)
         { case 0:a[i]++;break;
           case 1:a[i]--;
         }break;
    case 1:a[i]=0;
  }
  for(i=0;i<4;i++) printf("%d",a[i]); printf("\n");
}
```

A）3344　　B）2050　　C）3040　　D）0304

例题分析

本题主要考查 for 和 switch 语句结合起来的知识点，同时 swtich 语句之间套有 switch 语句，分析过程如下。

（1）当 i=0 的时候，执行：

```
case 0:switch(a[i]%2)
{case 0:a[i]++;break;
 case 1:a[i]--;
}break;
```

由于 a[0]%2=0，执行：

case0:a[i]++;break;

得到 a[0]的值为 3，跳出内部 switch 语句，但由于外部也有一个 break 语句，所以直接指向外面 for 循环的 i++，使 i 的值增加 1，执行 i=1。

（2）当 i=1 的时候，执行“case 1:a[i]=0;”得到 a[1]为 0，继续执行 for 循环的 i++，使 i 的值变为 2。

（3）当 i 等于 2 的时候，执行：

```
case0:switch(a[i]%2)
{case 0:a[i]++;break;
 case 1:a[i]--;
}break;
```

由于 a[2]的值为 5，执行“case1:a[i]--;”得到 a[2]为 4。

（4）当 i 的值为 3 的时候，执行“case1:a[i]=0;”得到 a[3]为 0，循环结束。

经过以上分析，此题选 B。

【例题 4】题目：有 1，2，3，4 这 4 个数字，能组成多少个互不相同且无重复数字的三位数？都是多少？

例题分析

可填在百位、十位、个位的数字都是 1，2，3，4。组成所有的排列后再去掉不满足条件的排列。

```
#include  "stdio.h"
void main()
{
   int i,j,k;
   printf("\n");
   for(i=1;i<5;i++)        /*以下为三重循环*/
```

```
    for(j=1;j<5;j++)
    for (k=1;k<5;k++)
    {
      if(i!=k&&i!=j&&j!=k)      /*确保 i，j，k3 位互不相同*/
      printf("%d,%d,%d\n",i,j,k);
         }
}
```

9.3 while 语句和其构成的循环结构

1. 一般格式

while（循环继续条件）

{ 循环体语句组；}

如果循环体语句组只由一条语句组成，可省略大括号{}。

2. 执行流程

执行流程如图 9-2 所示。

（1）求解“循环继续条件”表达式，如果其值为非 0（真的），转（2），否则转（3）；

（2）执行循环体语句组，然后转（1）；

（3）执行 while 语句的下一条语句。

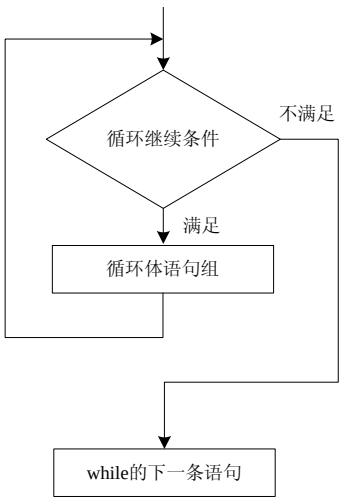


图 9-2 while 循环结构流程

说明：

（1）while 语句与 for 语句的不同之处在于 while 语句的变量初值语句放在了 while 语句的外面。

（2）while 语句的花括号是必须要有的，这与里面是否只有一条语句无关。

【例题 5】使用 while 语句改写【例题 1】中的程序。

例题分析

使用 while 语句编写与使用 for 结构是一样的。

```
#include "stdio.h"
void main()
{ int i,sum=0;
  i=1;
  while(i<=100)
 { if(i%3==0)
   sum=sum+i;
   i++;
 }
  printf("%d",sum);
}
```

【例题 6】设变量已正确定义，以下不能统计出一行中输入字符个数（不包含回车符）的程序段是（　　）。（2009 年 3 月）

A）n=0;while((ch=getchar())!='\n')n++;　　B）n=0;while(getchar()!='\n')n++;

C）for(n=0;getchar()!='\n';n++)　　D）n=0;for(ch=getchar();ch!='\n';n++);

例题分析

此题是将 getchar()语句与循环结构结合起来考查，由于 D 选项中的 ch=getchar();语句只能执行一次，不能达到题目要求。

9.4　do while 语句和其构成的循环结构

1. 一般格式

```
do
{ 循环体语句组；}
```

while（循环继续条件）;

当循环体语句组仅由一条语句构成时，可以省略花括号{}。

2. 执行流程

执行流程如图 9-3 所示。

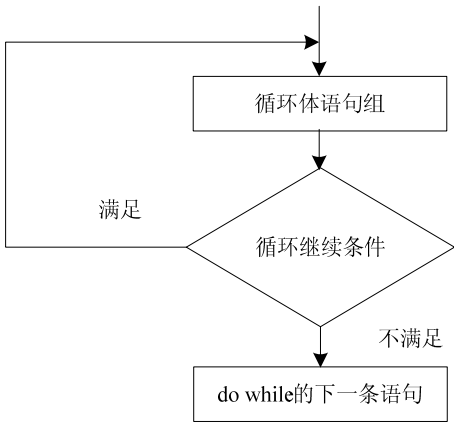


图 9-3　do while 循环结构流程图

（1）执行循环体语句组。

（2）计算“循环继续条件”表达式。如果表达式的值为非 0（真的），则转向（1）继续执行；否则，转向（3）。

（3）执行 do while 的下一条语句

说明：

（1）do while 语句中 while()后的分号不能省略，否则出现语法错误。

（2）do while 循环语句的特点是：先执行循环体语句组，然后再判断循环条件，所以 do while 循环至少会执行一次循环体。

（3）do while 语句，是当循环条件不成立时结束循环。

（4）for 语句也可与 while、do while 语句相互嵌套，构成多重循环。以下形式都是合法的嵌套。

①for(){…

```
    while()
    {…}
    …
     }
```

②do{

```
   …
   for()
   {…}
    …
    }while();
```

③while(){

```
  …
  for()
  {…}
  …
      }
```

④for(){

```
    …
    for(){
```

```
        …
            }
        }
```

【例题 7】使用 do while 语句改写【例题 1】中的程序。

例题分析

使用 do while 语句编写与前面的两种结构不同。

```
#include  "stdio.h"
void main()
{ int i,sum=0;
  i=1;
  do
  { if(i%3==0)
    sum=sum+i;
    i++;
  } while(i<=100);
  printf("%d",sum);
}
```

【例题 8】以下程序运行后的输出结果是_____。（2009 年 9 月)

```
#include <stdio.h>
main()
{ int a=1,b=7;
  do {
      b=b/2;a+=b;
     } while(b>1);
  printf("%d\n",a);}
```

例题分析

本题主要考查 do while 语句的使用，当 b 等于 7 的时候，执行 b=b/2，结果为 3，然后执行 a+=b，得到 a=4，由于 b>1，继续执行循环，直到 b<=1，循环结束，本题答案为 a=5。

9.5 循环语句的嵌套

一个循环语句的循环体内包含另一个完整的循环结构，称为循环的嵌套。循环的嵌套一般是 2 个到 3 个循环语句的联合使用。

【例题 9】有以下程序：（2009 年 9 月)

```
#include  "stdio.h"
main()
{ int f,f1,f2,i;
  f1=0;f2=1;
  printf("%d %d",f1,f2);
  for(i=3;i<=5;i++)
```

```
    { f=f1+f2;
      printf("%d",f);
      f1=f2;
      f2=f;
    }
    printf("\n");
}
```

例题分析

此题考查对 for 结构的理解，在这里关键是要搞清楚 f=f1+f2;f1=f2;f2=f 之间的赋值关系，这样就不会搞错了。输出 f 的结果是 7。

【例题 10】求出满足所有 x*x+y*y=100 的 x 和 y 的解。

例题分析

很显然，需要先求出 x 和 y 的范围，然后再利用穷举法把所有的 x 和 y 都带进方程中，运算一次，如果满足条件的就输出，从题目得知 x 的范围在[-10,10]之间，y 的范围在[-10,10]之间，所以采用如下的方法：

```
#include  "stdio.h"
void main()
{int x,y;
 for(x=-10;x<=10;x++)
   for(y=-10;y<=10;y++)
      if(x*x+y*y==100)
        printf("x=%d,y=%d",x,y);
}
```

【例题 11】求 1+2/3+3/5+4/7+5/9+…的前 20 项之和。

例题分析

（1）总体来说这是一个累加问题。

（2）这里的每个累加项是个分数，由分子与分母组成。

（3）分子的规律是自然数 1，2，3，…分母的规律是奇数 1，3，5，7，…

```
#include  <stdio.h>
void main()
{ int i,b=1;       /* i 为分子，b 为分母 */
    double s=0;     /* s 是累加器，并清 0，是双精度类型 */
    for(i=1;i<=20;i++)
       {  s+=(double)i/(double)b;
          b=b+2;
       }
     printf("s=%f\n",s);
}
```

【例题 12】求 Fibonacci 数列的前 40 个数。该数列的生成方法为：F1=1，F2=1，Fn=Fn-1+Fn-2（n≥3），即从第 3 个数开始，每个数等于前 2 个数之和。

例题分析

根据题意，设 F1=1，F2=1，

则 F3=F2+F1=1+1=2

F4=F3+F2=2+1=3
F5=F4+F3=3+2=5
F6=F5+F4=5+3=8

从求解 F3 的表达式可以看出，求出 F3 后就不再需要 F1 了，因此可以将 F3 的值暂时存入 F1，即 F2+F1=>F1(F3)，则求出 F4 的表达式也可改为 F4=F3+F2=F1+F2。

显然，求出 F4 后，也不再需要 F2 的值了，因此可以把 F4 的值暂时存入 F2 中，即 F1+F2=>F2(F4)。

依次类推，每一组数均可表示为：F1=F2+F1，F2=F1+F2。

【例题代码】

```
#include  <stdio.h>
void main()
{ long int f1=1,f2=1;      /* 定义并初始化数列的头两个数 */
  int i=1;                 /* 定义并初始化循环控制变量 i */
  for( ; i<=20; i++ )       /* 1 组 2 个数，20 组 40 个数 */
  { printf("%15ld%15ld", f1, f2);  /* 输出当前的 2 个数 */
    if(i%2==0)
       printf("\n");             /* 输出 2 次（4 个数），换行 */
    f1+=f2; f2+=f1;            /* 计算下两个数 */
  }
}
```

【例题 13】在 6～5000 内找出所有的亲密数对。若 a，b 是一对亲密数对，则 a 的因子和等于 b，b 的因子和等于 a，且 a 不等于 b。如 220、284 是一对亲密数对。

例题分析

（1）用试探法实现：只要会判断一个数（a）是否能找到它的亲密数就能求出所有的亲密数对了，用外层循环控制 6~5000 的数即可。

（2）设 sum1 为 a 的因子和，求出 a 的因子和。

（3）先满足“a 的因子和等于 b”这个条件，即 b=sum1，然后计算 b 的因子和 sum2。

（4）如果再满足条件“b 的因子和等于 a，且 a 不等于 b”，则 a，b 为一对亲密数对。

【例题代码】

```
#include  <stdio.h>
void main()
{ int a,b,c,k;
  int sum1,sum2;
  for(a=6; a<5001; a++)   /* 外循环：控制 6~5000 的数 */
  { sum1=0;
    for(c=1;c<=a/2;c++)
       if(a%c==0)
         sum1+=c;
    b=sum1;
    sum2=0;
    for(k=1;k<=b/2;k++)
       if(b%k==0)
          sum2+=k;
       if(sum2==a && a!=b)
```

```
        printf("%6d%6d\n",a,b);
    }
    }
```

9.6　break 和 continue 语句

为了使循环控制更加灵活，C 语言允许在特定条件成立时，使用 break 语句强行结束循环，或使用 continue 语句跳过循环体其余语句转向循环继续条件的判定。虽然 break 是非结构化语句（结构化语句只能有一个出口，而 break 的使用导致有多个出口），但使用 break 语句有时可以使程序变得简洁明了，要正确使用 break 和 continue 语句来中断或跳过某些循环。

1. 一般格式

```
break;
continue;
```

2. 功能

（1）break：强行结束循环，转向执行循环语句的下一条语句，即跳过整个循环语句。

（2）continue：对于 for 循环，跳过循环体其余语句，转向“循环变量增值”表达式的计算；对于 while 和 do while 循环，跳过循环体其余语句，但转向“循环继续条件”的判定，即跳过本次循环。

break 和 continue 语句对循环控制的影响如图 9-4、图 9-5、图 9-6 所示。

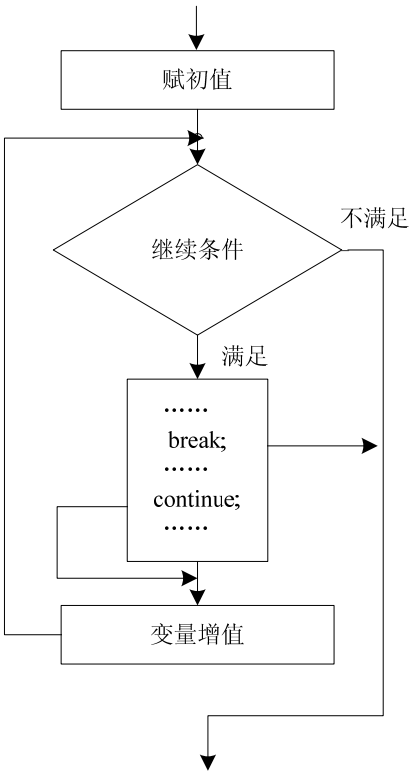


图 9-4　for 语句中的 break 和 continue 语句的执行流程

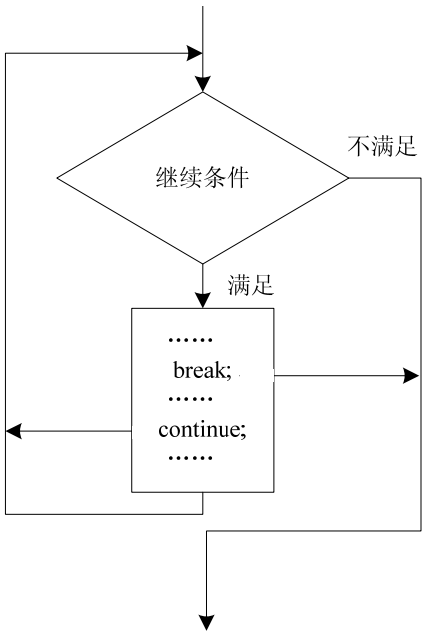


图 9-5　while 语句中 break 和 continue 语句的执行流程

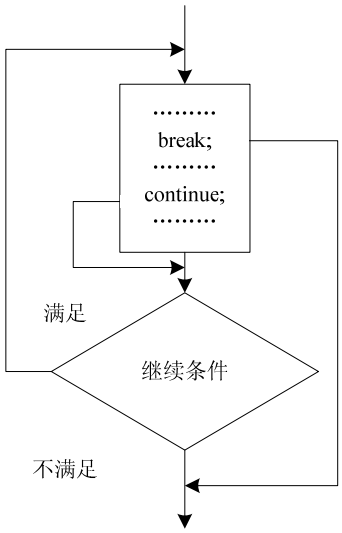


图 9-6　do while 语句中 break 和 continue 语句执行流程

说明：

（1）break 能用于循环语句和 switch 语句中，continue 只能用于循环语句中。

（2）循环嵌套时，break 和 continue 只影响包含它们的最内层循环，与外层循环无关。

【例题 14】利用 continue 语句完成输出 100 以内能被 7 整除的数。

例题分析

本例中，对 7～100 的每一个数进行测试，如该数不能被 7 整除，即余数运算不为 0，则由 continus 语句转去下一次循环。只有余数运算为 0 时，才能执行后面的 printf 语句，输出能被 7 整除的数。

```
#include  "stdio.h"
   void main(){
   int n;
   for(n=7;n<=100;n++)
     {
     if (n%7!=0)
     continue;
     printf("%d ",n);
   }
}
```

【例题 15】输出 100 以内的素数。

例题分析

所谓素数是指只能被 1 和自己整除的数，而整除表示的就是%运算符。在本例程序中，第一层循环表示对 1～100 这 100 个数逐个判断是否是素数，共循环 100 次，并设置一个标志 flag，在第二层循环中则对数 n 用 2～n-1 逐个去除，若某次除尽则跳出该层循环，并将 flag 设置为 0，说明不是素数。

```
#include  "stdio.h"
void main()
{
   int n,i;
```

```
    for(n=2;n<=100;n++)
      {flag=1;
       for(i=2;i<n;i++)
         if(n%i==0)
            {flag=0;
             break;
      }
    if(flag==1) printf("\n%d",n);
}
```

9.7　习题

9.7.1　选择题

1. 要求通过 while 循环不断读入字符，当读入字母 N 时结束循环。若变量已正确定义，下列选项中正确的程序段是（　　）。

A）while((ch=getchar())!='N')printf("%c",ch);

B）while(ch=getchar()!='N')printf("%c",ch);

C）while(ch=getchar()=='N')printf("%c",ch);

D）while((ch=getchar())=='N')printf("%c",ch);

2. 设变量已正确定义，则以下能正确计算 f=n！的程序是（　　）。

A）f=0
　for（i=1;i<=n;i++）f*=i;

B）f =1
　for（i=1;i<n;i++）f*=i;

C）f =1
　for（i=n;i>1;i++）f*=i;

D）f =1
　for（i=n;i>=2;i--）f*=i;

3. 有以下程序：

```
#include <stdio.h>
main()
{ int i,j;
  for(j=10;j<11;j++)
   {for(i=9;i<j;i++)
      if(j%i)break;
       if(i==j-1)
       printf("%d",j);
   }
}
```

输出结果是（　　）。

A）11　　B）10　　C）9　　D）10 11

4. 有以下程序:

```
#include <stdio.h>
main()
{ int k=5;
```

```
   while(--k)printf("%d",k-=3;);
   printf("\n")
}
```

A）1　　B）2　　C）4　　D）死循环

5. 当执行以下程序段时（　　）。

```
x=-1
do
{x=x*x;}
while(!x);
```

A）循环体将执行一次　　B）循环体将执行两次

C）循环体将执行无限次　　D）系统将提示有语法错误

6. 执行语句“for(i=1;i++<4;);”后，变量 i 的值是（　　）。

A）3　　B）4　　C）5　　D）不定

7. 有以下程序：

```
#include <stdio.h>
main()
{int x=4,y;
 do
   { y=x
     if(!y)printf("x");
     else
       printf("y");
     x--;
   }while(x);
}
```

程序的输出结果是（　　）。

A）xyyx　　B）yyyy　　C）yyxx　　D）yxyx

8. 下列程序的输出结果是（　　）。

```
#include <stdio.h>
main()
{ int i;
  for(i=1;i<=10;i++)
    { if((i*j>=20)&&(i*j<=100))
      break;
    }
  printf("%d\n",i*i);
}
```

A）49　　B）36　　C）25　　D）64

9. 有以下程序：

```
main()
```

```
{  int x=0,y=5,z=3;
   while(z-->0&&++x<5)
     y=y-1;
   printf("%d,%d,%d\n",x,y,z);
}
```

程序执行后的输出结果是（　　）。

A）3,2,0　　B）3,2,-1　　C）4,3,-1　　D）5,-2,-5

10. 有以下程序：

```
#include <stdio.h>
main()
{ int x,i;
  for(i=1;i<=50;i++)
  { x=i;
    if(x%2=0);
    if(x%3=0);
    if(x%7=0);
    printf("%d\n", i);
  }
 }
```

输出结果是（　　）。

A）28　　B）27　　C）42　　D）41

11. 有下列程序：

```
#include <stdio.h>
main()
{ int i,j,x=0;
  for(i=0,i<2;i++)
  { x++;
    for(j=0,j<=3;j++)
    { if(j%2)continue;
      x++;
    }
     x++;
    }
  printf("x=%d\n",x);
}
```

程序执行后的输出结果是（　　）。

A）x=4　　B）x=8　　C）x=6　　D）x=12

9.7.2　填空题

1. 下列程序的输出结果是_____。

```
 #include  "stdio.h"
main()
```

```
{  int i,sum=0;
   for(i=1;i<10;i++)
     if(i%3!=0)
      sum=sum+i;
   printf("sum=%d",sum);
}
```

2. 下列程序的意图是_____。

```
#include  "stdio.h"
main()
{  int i,a,b,c,d;
   for(i=1000;i<=5000;i++)
    {  a=i/1000;
       b=i%1000/100;
       c=i%100/10;
       d=i%10;
       if(a+b==c+d)
         printf("%d",i);
    }
}
```

3. 下列程序的结果是_____。

```
#include  "stdio.h"
main()
{  int i,j;
   for(i=1;i<3;i++)
    for(j=1;j<3;j++)
      if((i+j)%2==1)
        printf("%d,%d\n",i,j);
}
```

9.7.3 程序设计题

1. 一个球从 20 米高处落下并反弹，且每次反弹的高度为下落高度的 2/3。问第 3 次落地时共经过了多少米？第 3 次反弹的高度是多少？

2. 打印九九乘法表。

3. 输入一批正整数，以‘0’作为结束标志，统计数据个数、累计和，求平均值，找出最大值和最小值。

4. 计算数列 1，-1/3！，1/5！，-1/7！，1/9！…的和，至某项的绝对值小于 $1e^{-5}$ 为止。

5. 输出满足条件 x*x+y*y+z*z=2000 的所有解的情况。

第10章 字符型的数据

本章主要介绍数值数组、字符数组的定义和应用，结合 C 语言教学及计算机等级考试二级 C 语言考试中的要求，具体如表 10-1 所示。

表 10-1　考试要求

考试知识点	重要性
一维数组的定义和应用	★★
二维数组的定义和应用	★★★
字符数组的定义和应用	★★★
字符串函数	★★

10.1　字符常量

在前面讲解了使用整型数据、浮点型数据，还有一种重要的数据在 C 语言程序中，那就是字符型数据。

字符类型的常量也可分为两类。

（1）一般普通的字符常量，那就是用一对单引号引起来的一个字母。如‘a’、‘M’、‘%’、‘*’等。

（2）特殊形式的字符常量，就是以字符“\”开头的用单引号引起来的字符常量，也称这样的字符常量为转义字符。引出这类字符的主要原因是这些字符难以用通常的方法表示，比如前面学到的换行符‘\n’，这个字符是显示不出来的，但是格式需要这样的控制。表 10-2 所示是转义字符表。

表 10-2　转义字符表

转义字符	含义	ASCII 代码
\a	产生警告，发出提示	7
\b	退格，表示光标移到前一列	8
\f	换页，光标移到下一页开头	12
\n	换行，光标移到下一行的开头	10
\r	回车，光标移到当前行的开头	13
\t	光标移到下一个制表符位置	9
\\	产生一个\符号	92
\’	产生一个‘符号	39
\”	产生一个“符号	34

（续表）

转义字符	含义	ASCII 代码
\0	产生一个空字符	0
\ddd	任意字符	三位八进制
\xhh	任意字符	二位十六进制

10.2 字符变量

字符变量只能存储一个字符，字符变量的定义关键字是 char。如：

char ch1;

ch1='a';

由于字符型数据存储的是字符的 ASCII 码，在存储形式上和整型类似，所以在 C 语言中整型数据和字符型数据可以通用，也就是说一个整型数据可以直接赋给一个字符型数据，一个字符型数据可以直接赋给整型数据。一个字符型数据可以按照整型数据类型输出，一个整型数据也可以按照字符型数据输出。

10.2.1 字符串常量

一个或者多个字符用双引号引起来就是字符串常量。如，

"what's you name.","2012 Olympic Games."

字符串的存储用一个特殊的字符来表示结束，这个字符就是‘\0’，这个字符会在计算机存储一个字符串常量时，自动加在字符串之后进行存储，所以字符串常量和字符常量是有区别的，如“a”，‘a’是有区别的。这里很明显就是‘a’存储时就是存储一个字符 a 的 ASCII 码值，而“a”存储时就不同了，除了存储字符 a 之外，还要在 a 之后存储一个字符‘\0’作为字符串的结束标志。

10.2.2 常用输出格式

%s 输出实际长度字符串。

%ms 输出的字符串占 m 位，如果串长度小于 m，左补空格，如果大于 m，实际输出。

%-ms 输出的字符串占 m 位，如果串长度小于 m，右补空格。

%m.ns 输出的字符串占 m 位，但只取字符串中左端 n 个字符并靠右对齐。

%-m.ns 其中 m 和 n 含义同上，靠左对齐，如果 n>m，则 m 自动取 n 值。

【例题 1】有以下定义语句，编译时会出现编译错误的是（　　）。（2009 年 9 月）

A）char a='a';　　B）char a='\n';　　C）char a='aa';　　D）char a='\x2d';

例题分析

根据字符型变量的定义，发现答案选的是 C。

【例题 2】有以下程序：（2009 年 9 月）

```
#include <stdio.h>
main()
{ char c1,c2;
  c1='A'+'8'-'4';
  c2='A'+'8'-'5';
  printf("%c,%d\n",c1,c2);
}
```

已知字母 A 的 ASCII 码为 65，程序运行后的输出结果是（　　）。

A）E,68　　B）D,69　　C）E,D　　D）输出无定值

例题分析

此题主要考查字符型变量在进行计算的知识点，c1 是以%c 的形式输出的，结果为 E，而 c2 是以%d 的形式输出的，结果为 68，所以，此题答案选 A。

10.3　字符的输入和输出

前面讲过单个字符的输入和输出，在这里介绍另外一种方式关于字符的输入和输出。

10.3.1　采用 scanf()语句

由于是字符型变量的输入，所以在输入格式上要采用%c 的形式，即：

scanf（“%c%c……%c”，&变量名 1，[&变量名 2]，……，[&变量名 n]）；

10.3.2　采用 printf()语句

由于输出的是字符型变量，所以输出格式上采用%c 的形式，即：

printf（“%c%c……%c”，变量名 1，[变量名 2]，……，[变量名 n]）；

10.4　一维数组的定义和一维数组元素的引用

在 C 语言中，数组属于构造数据类型。一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或是构造类型，因此按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组和结构数组等各种类别。

10.4.1　数组的定义

1. 数组的定义

数组在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。数组是由数组类型、数组名和长度构成的。

2. 数组类型说明

（1）数组类型实际上是指数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。

（2）数组名的书写规则应符合标识符的书写规定。

（3）数组名不能与其他变量名相同。

（4）方括号中常量表达式表示数组元素的个数，如 a[5]表示数组 a 有 5 个元素。但是其元素从 0 开始计算，因此 5 个元素分别为 a[0]，a[1]，a[2]，a[3]，a[4]。

（5）不能在方括号中用变量来表示元素的个数，但是可以是符号常数或常量表达式。

（6）允许在同一个类型说明中，说明多个数组和多个变量。

10.4.2　一维数组的定义

1. 一维数组的定义

类型说明符 数组名 [n]；

其中：

类型说明符，可以是 C 语言中允许的任何类型。例如，int，float 或 char。

数组名是一个标识符，表示定义的数组的名字。

n 表示数组中的元素个数，也称为数组的长度，它可以是任何的整型表达式，这当然包括整型常量和符号常量。我们可以用数组的下标值来标识某个数组的元素。

说明：

（1）C 语言中，数组下标总是从 0 开始的。即各元素的下标是从 0 开始，至 n-1 为止的整数。

（2）数组的类型实际上是指数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。

（3）数组名的书写规则应符合标识符的书写规定。

（4）数组名不能与其他变量名相同。

（5）其中下标只能为整型常量或整型表达式。

例如 int a[5]，如图 10-1 所示。

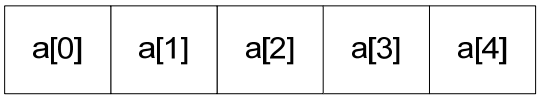


图 10-1　一维数组存储

2. 一维数组的初始化

格式：类型说明符 数组名={常量列表}；

功能：在定义数组时对数组元素赋以初值。

说明：

（1）常量列表可以是数值型、字符常量或字符串。

（2）数组元素的初值必须依次放在一对花括号内。

例如：

（1）给数组 a 各元素赋以初值：

int a[10]={0,1,2,3,4,5,6,7,8,9};

（2）可以只给一部分元素赋初值，后几个元素值为 0。

int a[10]={0,1,2,3,4,5};花括弧内只有 6 个初值，系统默认给前 6 个元素赋初值，后 4 个元素值为 0。

（3）如果想使一个数组中全部元素值为 0，可以写成：

int a[10]={0,0,0,0,0,0,0,0,0,0};

（4）给全部数组元素赋初值时，可以不指定数组长度：

int a[6]={1,2,3,4,5,6};

可以写成：int a[ ]={1,2,3,4,5,6};

【考生注意】

如果被定义的数组长度与花括号内提供的初值个数不一样时，则数组长度不能省略。如 int a[10]={0,1,2,3,4,5}。

10.4.3　一维数组的引用

数组元素的一般形式为：数组名[下标]

由于在输入和输出数组元素时，经常采用 for 语句的形式来表示。以 a[N]为例（N 是一个具体的数值）。

```
for(i=0;i<N;i++)
  scanf("%d",&a[i]);
for(i=0;i<N;i++)
  printf("%d",a[i]);
```

【例题 3】写出下列程序的结果。（2009 年 9 月）

```
#include <stdio.h>
main()
{ char s[]={"012xy"};int i,n=0;
  for(i=0;s[i]!=0;i++)
     if(s[i]>='a'&&s[i]<='z') n++;
```

```
  printf("%d\n",n);
}
```

程序运行后的输出结果是（　　）。

A）0　　B）2　　C）3　　D）5

例题分析

此题主要考查一维数组的使用，此题的意思是统计数组 s 中小写字母的个数，此题的答案选 B。

10.5　一维数组的应用举例

【例题 4】输入 10 位学生的成绩，求出平均分，并输出高于平均分的同学的成绩。

```
#include  "stdio.h"
void main()
{ int i;
  float score[10],aver=0.0;
  printf("Please input scores of 10 students:");
  for(i=0;i<10;i++)   /*输入 10 位学生成绩并累加和*/
  {  scanf("%f",&score[i]);
     aver+=score[i];
  }
  aver/=10;       /*求出 10 位学生的平均成绩*/
  printf("The average score is:%.2f\n",aver);
  printf("They are:");
  for(i=0;i<10;i++)   /*输出高于平均成绩的学生成绩*/
     if(score[i]>aver)
         printf("%6.2f",score[i]);
}
```

程序运行结果：

Please input scores of 10 students: 80　90　68　67　89　45　78　85　86　66

The average score is : 75.4

They are:　80.00　90.00　89.00　85.00　86.00

【例题 5】用冒泡法对 10 个整型数按升序排序。

冒泡法是使较小的值像空气泡一样逐渐“上浮”到数组的顶部，而较大的值逐渐下沉到数组的底部。

具体思路是：从第一数开始将相邻的两个数进行比较，较大的数向后移动，较小的数“上浮”一个，经过一轮比较，最大的数移动到末尾。对剩下的数继续下一轮的比较和移动。如果 n 个数比较，这样 n-1 轮后，就完成了排序工作。程序如下：

```
#include  "stdio.h"
void main()
{  int i, j, t, a[10];
   printf("Please input 10 numbers:\n");
   for(i=0;i<10;i++)       /*输入 10 个整数存入数组 a 中*/
      scanf("%d",&a[i]);
```

```
    for(i=0;i<9;i++)        /*对数组 a 中的 10 个整数排序*/
    for(j=0;j<9-i;j++)
      if(a[j]>a[j+1])       /*前面的元素大于后面的元素则交换*/
      {  t=a[j];    a[j]=a[j+1];    a[j+1]=t;  }
    printf("The sorted numbers are:");
    for(i=0;i<10;i++)       /*输出数组 a 中的 10 个元素*/
      printf("%2d ",a[i]);
    printf("\n");
}
```

程序运行结果：

Please input 10 numbers：

3，5，1，7，9，8，0，6，2，4↙

The sorted numbers are：

1 2 3 4 5 6 7 8 9

具体的执行步骤如下：

第一趟第 1 次比较结果：

{3，5}，1，7，9，8，0，6，2，4

第一趟第 2 次比较结果：

3，{1，5}，7，9，8，0，6，2，4

第一趟第 3 次比较结果：

3，1，{5，7}，9，8，0，6，2，4

第一趟第 4 次比较结果：

3，1，5，{7，9}，8，0，6，2，4

第一趟第 5 次比较结果：

3，1，5，7，{8，9}，0，6，2，4

第一趟第 6 次比较结果：

3，1，5，7，8，{0，9}，6，2，4

第一趟第 7 次比较结果：

3，1，5，7，8，0，{6，9}，2，4

第一趟第 8 次比较结果：

3，1，5，7，8，0，6，{2，9}，4

第一趟第 9 次比较结果：

3，1，5，7，8，0，6，2，{4，9}

经过第一趟的比较把最大的一个数 9 排到了最后，接下来进行第二趟比较，方法也类似，同样把次大的一个数 8 排到了 9 之前，直到第 9 趟排序结束为止。若有 n 个数，就要比较 n-1 趟。

10.6 二维数组的定义和二维数组元素的引用

10.6.1 二维数组的定义

1. 二维数组的定义

类型说明符 数组名[常量表达式 1][常量表达式 2]

常量表达式 1 是数组元素的行数，常量表达式 2 是数组元素的列数。二维数组在概念上是二维的，即是说其下标在两个方向上变化，下标变量在数组中的位置也处于一个平面之中，而不是像一维数组只是一个向量。但是，实际的硬件存储器却是连续编址的，也就是说存储器单元是按一维线性排列的。如何在一维存储器中存放二维数组，可有两种方式：一种是按行排列，即放完一行之后顺次放入第二行；另一种是按列排列，即放完一列之后再顺次放入第二列。在 C 语言中，二维数组是按行排列的。

说明：

（1）二维数组可看做特殊的一维数组。如 int a[2][3]，我们可以把数组 a 看做一个一维数组，它有两个元素，分别为 a[0]，a[1]，每个元素又由具有 3 个元素的一维数组组成。

（2）数组元素在内存排列顺序为按行存放。如 int a[2][3]，它在内存中的存储情况如图 10-2 所示。

a[0][0]
a[0][1]
a[0][2]
a[1][0]
a[1][1]
a[1][2]

图 10-2　二维数组 a[2][3]

2. 二维数组的初始化

定义二维数组时，给数组元素赋初值称为二维数组初始化。二维数组初始化时要注意二维数组的元素排列顺序。初始值的排列顺序必须与数组元素在内存的存储顺序完全一致。具体方法如下。

（1）按行给二维数组赋初值。

例如：“int a[2][3]={{1,2,3},{4,5,6}};”按行赋初值的方法直观。

（2）按数组排列顺序对各元素赋初值。

例如：“int a[2][3]={1,2,3,4,5,6};”，此方法数据所处的行列位置不直观，尤其是数据多时，数据存储所在的行列需要仔细定位，否则容易出现错误。

（3）对部分元素赋初值。

例如：“int a[2][3]={{1,1},{2,4}};”此法对数组中各行部分元素赋初值，其余元素值自动为 0，即赋值后数组 a 的各元素为：

```
1    1    0
2    4    0
```

（4）赋初值时，有些情况可缺省常量表达式 1 的长度，但常量表达式 2 的长度不能缺省。

如为数组的全部元素都赋初值时，则定义数组时常量表达式 1 可以缺省。

例如：“int a[ ][3]={1,2,3,4,5,6};”等价于“int a[2][3]={1,2,3,4,5,6};”，系统可以根据花括号内的元素个数与列下标的关系，求出行下标的值为 2。

10.6.2　二维数组的引用

数组名[行下标表达式][列下标表达式]

二维数组元素的输入和输出和一维数组的输入和输出是一样的，在一维数组中采用一个 for 语句，而二维数组采用两个 for 语句，int a[N][M]（N 和 M 表示一个具体的数值）。

```
for（i=0;i<N;i++）
  for（j=0;j<M;j++）
    scanf（"%d",&a[i][j]）;
for（i=0;i<N;i++）
  for（j=0;j<M;j++）
    printf（"%d",a[i][j]）;
```

说明：

（1）“行下标表达式”和“列下标表达式”都应是整型表达式或符号常量。

（2）“行下标表达式”和“列下标表达式”的值，都应在已定义数组大小的范围内。

【例题 6】二维数组输入与输出方法示例。

```
#include  "stdio.h"
void main()
{
   int i, j, a[3][4];
   printf("input array numbers:\n");
   for(i=0;i<3;i++)
     for(j=0;j<4;j++)
         scanf("%d",&a[i][j]);
   printf("output array numbers:\n");
   for(i=0;i<3;i++)
   { for(j=0;j<4;j++)
        printf("%d",a[i][j]);
      printf("\n");
   }
}
```

程序运行情况：

input array numbers:

1 2 3 4 5 6 7 8 9 10 11 12↙

output array numbers:

1 2 3 4

5 6 7 8

9 10 11 12

10.7 二维数组应用举例

【例题 7】从键盘为一个 3×3 整型数组输入数据，并找出主对角线上元素的最大值及其所在的行号。

例题分析

本程序中对角线上的元素只要满足行号与列号相等就是本题所要找的答案

```
#include  "stdio.h"
void main( )
{ int a[3][3],i, j, max, row;
  for(i=0;i<3;i++)
  for(j=0;j<3;j++)
     scanf("%d",&a[i][j]);
  max=a[0][0];  /*设 a[0][0]为最大的元素*/
  row=0;      /*设第 0 行为最大元素所在的行号*/
  for(i=1;i<3;i++)
    if(max<a[i][i])
    { max=a[i][i];
      row=i;
```

```
        }
    printf("max=%d,row=%d",max,row);
}
```

【例题 8】设一个二维数组 a[3][4]存放 3 个人 4 门课的成绩，再设一个一维数组 v[4]存放所求得各分科平均成绩，设变量 aver 为全组各科总平均成绩。

```
#include  "stdio.h"
void main()
{
   int i,j,v[4],a[3][4];
   float aver,s=0.0;
   printf("input score\n");
   for(i=0;i<3;i++)
   {
     for(j=0;j<4;j++)
     { scanf("%d",&a[i][j]);
       s=s+a[i][j];
     }
      v[i]=s/4.0;
      s=0;
      }
      aver=(v[0]+v[1]+v[2]+v[3])/3;
      printf("%d,%d,%d,%d\n",v[0],v[1],v[2],v[3]);
      printf("total:%f\n",aver);
}
```

10.8　字符串

在 C 语言的字符应用中，有两种方式，一种是字符数组，一种是字符串。这两种方式的使用占据 C 语言字符类型数据使用的大部分方面。

10.8.1　字符数组的定义

字符数组是用来存放字符数据的数组，即数组的数据类型是字符型（char）的数组称为字符数组。字符数组的每个元素存放一个字符。

一维数组的定义形式：　char　数组名[常量表达式];

如：char　a[10];

二维数组的定义形式：　char　数组名[常量表达式 1][常量表达式 2];

如：char　a[3][4];

用来存放字符数据的数组是字符数组。字符数组中的一个元素存放一个字符。

10.8.2　字符数组的初始化

字符数组的初始化就是把字符赋值给数组各元素。

（1）一维字符数组初始化，如，char c[10]={ 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}。

（2）如果花括号中提供的数值个数（即字符个数）大于数组长度，则做语法错误处理。

（3）如果初值个数小于数组长度，则只将这些字符赋给数组中前面的那些元素，其余元素自动赋给为空字符（‘\0’）。

如，char c[2][10]={{ 'a', 'b', 'c'},{'d', 'e', "}};

字符数组 c 各元素初值为：{'a' , 'b' , 'c' , '\0, ' '\0' , '\0' , '\0' , '\0' , '\0' , '\0'}

{'d' , 'e' , " , '\0' , '\0' , '\0' , '\0' , '\0' , '\0' , '\0'}

（4）当初值个数与字符数组长度相同，在定义时可以省略数组长度，系统会自动根据初值个数确定数组长度。

10.8.3 字符数组的引用

字符数组的逐个字符引用，与前面所讲的数组引用类似。

10.8.4 字符串的处理

1. 字符串和字符串结束标志

（1）字符串常量。字符常量是用双引号括起来的一串字符，C 语言约定用‘\0’作为字符串的结束标志，它占内存空间，同时计入串的长度。

（2）在 C 语言中，字符串可以存放在字符型一维数组中，故可以用字符型一维数组处理字符串。

2. 用字符串常量给数组赋初值（初始化）

char c[6]="china";

说明：

（1）如果提供的字符个数大于数组长度，系统报错。

（2）如果提供的字符个数小于数组长度，则在最后一个字符后加‘\0’作为字符串结束标志。

【考生注意】

如果考试题目中询问数组的实际长度，不要把‘\0’算进去，如果询问数组在内存中所占的长度，那就要把‘\0’计算在内。

【例题 9】判断一个字符串是否为回文串（回文串是指正读反读都一样的字符串，如字符串"abc121cba"）。

```
#include  "string.h"
#include  "stdio.h"
void main()
{
   char x[20];
   int i,j,n;
   gets(x);
   n=strlen(x);
   i=0;j=n-1;
   while(x[i]==x[j]&&i<j)
     {i++;j--;}
   if(i>=j)
     printf("The number is palindrome");
   else
     printf("The number is not palindrome");
}
```

10.9　字符串输入和输出

10.9.1　输入字符串 gets()函数

（1）调用方式：gets（字符数组）。

（2）函数功能：从标准输入设备——键盘上，读取 1 个字符串（可以包含空格），并将其存储到字符数组中去。

说明：

1）gets()读取的字符串，其长度没有限制，编程者要保证字符数组有足够大的空间存放输入的字符串。

2）该函数输入的字符串中允许包含空格，而 scanf()函数不允许。

【例题 10】写出下列程序的结果。

```
#include  "stdio.h"
main()
{
   char c[80];
   gets(c);
}
```

10.9.2　输出字符串 puts()函数

功能：将一个字符串输出到终端，该函数返回值是字符数组的起始地址。

（1）调用方式：puts（字符数组）。

（2）函数功能：把字符数组中所存放的字符串，输出到标准输出设备中去，并用‘\n’取代字符串的结束标志‘\0’。所以用 puts()函数输出字符串时，不要求另加换行符。

（3）使用说明

1）字符串中允许包含转义字符，输出时产生一个控制操作。

2）该函数一次只能输出一个字符串，而 printf()函数也能用来输出字符串，但一次能输出多个。

【考生注意】用 puts 和 gets 函数只能输入或输出一个字符串，不能写成“puts(str1,str2);”或“gets(str1,str2);”。

【例题 11】写出下列程序的结果。

```
#include  "stdio.h"
main()
{
   char c[]="I Love China";
   puts(c);
}
```

【例题 12】输入 10 位学生的成绩，求出平均分，并输出高于平均分的同学成绩。

```
#include  "stdio.h"
void main()
{ int i;
   float score[10],aver=0.0;
   printf("Please input scores of 10 students:");
   for(i=0;i<10;i++)   /*输入 10 位学生成绩并累加和*/
```

```
    { scanf("%f",&score[i]);
      aver+=score[i];
     }
     aver/=10;      /*求出 10 位学生的平均成绩*/
     printf("The average score is:%.2f\n",aver);
     printf("They are:");
     for(i=0;i<10;i++)  /*输出高于平均成绩的学生成绩*/
       if(score[i]>aver)
          printf("%6.2f",score[i]);
}
```

10.10 字符串处理函数

字符串处理函数主要用来对字符串进行各种运算操作。

10.10.1 字符串比较 strcmp()函数

（1）调用方式：strcmp（字符串 1，字符串 2）。

其中“字符串”可以是串常量，也可以是一维字符数组。

（2）函数功能：比较两个字符串的大小。

如果：字符串 1=字符串 2，函数返回值等于 0；

字符串 1<字符串 2，函数返回值负整数；

字符串 1>字符串 2，函数返回值正整数。

（3）使用说明

1）如果一个字符串是另一个字符串从头开始的子串，则母串为大。

2）不能使用关系运算符= =来比较两个字符串，只能用 strcmp() 函数来处理。

【例题 13】写出下列程序的结果。

```
#include  "string.h"
void main()
{ int k;
  static char st1[15],st2[]="C Language";
  printf("input a string:\n");
  gets(st1);
  k=strcmp(st1,st2);
  if(k==0) printf("st1=st2\n");
  if(k>0) printf("st1>st2\n");
  if(k<0) printf("st1<st2\n");
  printf("%d",k);
}
```

【例题 14】输入 3 个字符串，按英文字母顺序排列后输出。

```
#include  <string.h>
```

```
void main()
{
   char s1[10]="China", s2[10]="America", s3[10]="Japan", t[10];
   if(strcmp(s1,s2)>0)
     {strcpy(t,s1);strcpy(s1,s2);strcpy(s2,t);}
   /* 如果 strcmp(s1,s2)>0，交换 s1 与 s2 字符串*/
   if(strcmp(s2,s3)>0)
     {strcpy(t,s2);strcpy(s2,s3);strcpy(s3,t);}
   puts(s1);
   puts(s2);
   puts(s3);
}
```

【考生注意】对两个字符比较，不能用以下形式：

if(str1==str2)

printf("yes");

而只能用：

if(strcmp(str1,str2)==0)

printf("yes");

10.10.2　测试字符串长度函数 strlen（字符数组）

功能：测试字符串长度，函数值为字符串中的实际长度，不包括‘\0’在内。例如：

char str[10]={"china"};

printf("%d",strlen(str));　/*strlen 是不把‘\0’计入字符串长度的*/

结果为：5

10.10.3　字符串拷贝 strcpy()函数

（1）调用方式：strcpy（字符数组，字符串）。

其中“字符串”可以是串常量，也可以是字符数组。

（2）函数功能：将“字符串”完整地复制到“字符数组”中，字符数组中原有内容被覆盖。

说明：

1）字符数组必须定义得足够大，以便容纳复制过来的字符串。复制时，连同结束标志‘\0’一起复制。

2）不能用赋值运算符=将一个字符串直接赋值给一个字符数组，只能用 strcpy()函数来处理。

如：　char c1[30],c2="How are you!\n";

　　printf("%s ",strcpy(c1,c2));

　　输出结果：How are you!

3）可以用此函数将字符串中的若干个字符复制到字符数组中去。如：strcpy(str1,str2,3);

作用是把 str2 中的前 3 个字符复制到 str1 中去。

【例题 15】将字符串 st2 放到 st1 中去。

```
#include  "string.h"
void main()
{
```

```
    static char st1[15],st2[]="C Language";
    strcpy(st1,st2);
    puts(st1);printf("\n");
}
```

10.10.4 字符串连接 strcat()函数

（1）调用方式：strcat（字符数组，字符串）。

（2）函数功能：把“字符串”连接到“字符数组”中的字符串尾端，并存储于“字符数组”中。“字符数组”中原来的结束标志被“字符串”的第一个字符覆盖，而“字符串”在操作中未被修改。

说明：

1）由于没有边界检查，编程者要注意保证“字符数组”定义得足够大，以便容纳连接后的目标字符串；否则，会因长度不够而产生问题。

2）连接前两个字符串都有结束标志‘\0’，连接后“字符数组”中存储的字符串的结束标志‘\0’被舍弃，只在目标串的最后保留一个‘\0’。

例如：char c1[30]= "How are you!\n";

char c2[ ]= "I am fine!";

printf("%s ",strcat(c1,c2));

输出结果：How are you!

I am fine!

10.10.5 将字符串中大写字母转换成小写 strlwr()函数

（1）调用方式：strlwr（字符串）。

（2）函数功能：将字符串中的大写字母转换成小写，其他字符（包括小写字母和非字母字符）不转换。

10.10.6 将字符串中小写字母转换成大写 strupr()函数

（1）调用方式：strupr（字符串）。

（2）函数功能：将字符串中小写字母转换成大写，其他字符（包括大写字母和非字母字符）不转换。

【例题 16】简单密码检测程序。

例题分析

本程序是一个用来验证简单密码是否正确的程序，整个思想过程是：输入密码与已知密码 password 进行比较。假如正确，通过 break 语句来跳出循环；否则，则系统提示“口令错误，按任意键继续”，不过只有 3 次机会，超过 3 次的话系统自动退出程序。

```
#include  "stdio.h"
void main()
 {  char pass_str[80];     /*定义字符数组 pass_str*/
    int i=0;
    while(1) /*检验密码*/
    {
       printf("请输入密码:\n");
       gets(pass_str);                  /*输入密码*/
       if(strcmp(pass_str,"password")!=0)    /*口令错*/
```

```
        printf("口令错误，按任意键继续");
      else
        break;              /*输入正确的密码，终止循环*/
      i++;
      if(i==3) exit(0);    /*输入 3 次错误的密码，退出程序*/
   }
   /*输入正确密码所进入的程序段*/
      printf("密码输入成功! ");
}
```

【例题 17】将字符串 s1 从第 m 个字符开始剩余的所有字符，送入字符数组 s2 中。

例题分析

值得注意的是在复制字符串的同时，最后不要忘了给 s2[j]加上结束标志符‘\0’。

```
#include  "stdio.h"
void main()
{
   int i,j,m;
   char s1[80],s2[80];
   printf("input a string:\n");
   gets(s1);
   printf("input start point:\n");
   scanf("%d",&m);
   i=m-1; j=0;
   while(s1[i]!='\0')
   {s2[j]=s1[i];
    i++;
    j++;
   }
   s2[j]='\0';
   puts(s2);
}
```

10.11　习题

10.11.1　选择题

1. 以下能正确定义二维数组的是（　　）。

A）int　a[ ] [3];　　B）int　a[ ] [3]={2*3};

C）int　a[ ] [3]={};　　D）int　a[2] [3]={{1},{2},{3,4}};

2. 以下程序的输出结果是（　　）。

```
#include <stdio.h>
main()
{ int i,k,a[10],p[3];
```

```
   k=5;
   for(i=0;i<10;i++)a[i]=i;
   for(i=0;i<3;i++)p[i]=a[i*(i+1)];
   for(i=0;i<3;i++)k+=p[i]*2;
     printf("%d\n",k);
}
```

A）20　　B）21　　C）22　　D）23

3. 有以下程序：

```
#include  <stdio.h>
main()
{ int c;
  while((c=getchar())!='\n')
     { switch(c-'3')
        {  case 0:
           case 1:putchar(c+4);
           case 2:putchar(c+4);break;
           case 3:putchar(c+3);
           case 4:putchar(c+3);break;
        }
     }
  printf("\n");
}
```

从第一列开始输入以下数据（<CR>代表一个回车符）：3845<CR>，程序的输出结果是（　　）。

A）77889　　B）77868　　C）776810　　D）77886610

4. 有以下程序：

```
#include <stdio.h>
 main()
 { char   p[ ]={'a','b','c'},q[]="abc";
    printf("%d%d\n",sizeof(p),sizeof(q));
 }
```

程序运行后的输出结果是（　　）。

A）4　4　　B）3　 3　　C）3　4　　D）4　3

5. 在 C 语言中，char 型数据在内存中的存储形式为（　　）。

A）反码　　B）补码　　C）ASCII 码　　D）原码

6. 运行下面的程序，若从键盘输入字母“a”，则输出结果是（　　）。

```
char c;
c=getchar();
if(c>='a'&&c<='g')c=c+4;
else if(c>='g'&&c<='z')c=c-21;
     else printf("input error!\n");
putchar(c);
```

A）f　　B）t　　C）e　　D）d

7. 若已包括头文件<string.h>且已有定义 char s1[18]，s2={"ABCDE"}和 int i，现在将字符串“ABCDE”赋给 s1，下列语句错误的是（　　）。

A）strcpy(s1,s2)　　B）strcpy(s1,"ABCDE");

C）s1="ABCDE";　　D）for(i=0;i<6;i++)

10.11.2　填空题

1. 以下程序运行后的输出结果是_____。

```
#include "stdio.h"
main()
{ int a[4][4]={{1,2,3,4},{5,6,7,8},{11,12,13,14},{15,16,17,18}};
  int i=0,j=0,s=0;
  while(i++<4)
  {
    if(i==2||i==4)continue;
    j=0;
    do{s+=a[i][j];j++;}while(j<4);
  }
    printf("%d\n",s);
}
```

2. 下列程序中的数组 a 包括 10 个整数元素，分别将前项和后项之和存入数组 b，并按每行 4 个元素输出数组 b。请填空。

```
#include  <stdio.h>
main()
{ int a[10],b[10],i;
  for(i=0;i<10;i++)
     scanf("%d",&a[i]);
  for(i=0;i<9;i++)
     _____;
  for(i=0;i<9;i++)
  { if(i%4==0)printf("\n");
    printf("%3d",b[i]);
  }
}
```

10.11.3　程序设计题

1. 求若干个数（不多于 10 个）的最大数、最小数。

2. 求一个 5×5 矩阵的对角线之和。

3. 有 M 个学生，学习 N 门课程，已知所有学生的各科成绩，编程：分别求每个学生的平均成绩和每门课程的平均成绩。

第11章 函数

本章主要介绍函数的概念和应用，结合计算机等级考试中的要求，具体如表 11-1 所示。

表 11-1　考试要求

掌握知识点	重要性
函数的定义和调用	★★
函数嵌套调用	★★
函数递归调用	★★
数组作为函数参数的使用	★★★★

11.1　函数的定义

C 源程序是由函数组成的。虽然在前面各章的程序中都只有一个主函数 main()，但实用程序往往由多个函数组成。函数是 C 源程序的基本模块，通过对函数模块的调用实现特定的功能。C 语言中的函数相当于其他高级语言的子程序，C 语言不仅提供了极为丰富的库函数（如 Turbo C，MSC 都提供了三百多个库函数），还允许用户建立自己定义的函数。用户可把自己的算法编成一个个相对独立的函数模块，然后用调用的方法来使用函数。

可以说 C 程序的全部工作都是由各式各样的函数完成的，所以也把 C 语言称为函数式语言。由于采用了函数模块式的结构，C 语言易于实现结构化程序设计。使程序的层次结构清晰，便于程序的编写、阅读、调试。

在 C 语言中可从不同的角度对函数分类。

从函数定义的角度看，函数可分为库函数和用户定义函数两种

（1）库函数

由 C 系统提供，用户无须定义，也不必在程序中做类型说明，只需在程序前调用有该函数原型的头文件即可在程序中直接调用。在前面各章的例题中反复用到的 printf，scanf，getchar，putchar，gets，puts，strcat 等函数均属此类。

（2）用户定义函数

由用户按需要写的函数。对于用户自定义函数，不仅要在程序中定义函数本身，而且在主函数模块中还必须对该被调函数进行类型说明，然后才能使用。

任何函数（包括主函数 void main()）都是由函数说明和函数体两部分组成的。根据函数是否需要参数，我们可以把函数分为无参函数和有参函数两种类型。

1. 无参函数的一般形式

函数类型　函数名（void）

{

说明语句部分；
可执行语句部分；
}

函数类型指定函数带回来的值的类型。无参函数一般不需要带回函数值，因此我们可以不写函数类型。

2. 有参函数的一般形式

函数类型　函数名（数据类型　参数[数据类型　参数 1，数据类型　参数 2……] ）
{　说明语句部分；
可执行语句部分；
}

【考生注意】函数不可缺省参数表；如果不需要参数，则用 void 表示没有返回值。主函数 void main()除外。

11.2　函数的参数和返回值

在 C 语言中，除了主函数就是子函数了，平时除了主函数以外的函数都是子函数（函数），函数是由主函数进行调用的，那么主函数是如何调用子函数的呢？主要通过将参数从主函数传递给子函数，然后根据实际的情况返回相应的值。

11.2.1　函数的参数

函数的参数分为实际参数（实参）和形式参数（形参）两种，作用是实现两者的数据传送。

实参出现在主调用函数中，形参出现在函数定义中，只能在该函数体内使用。发生函数调用时，主调用函数把实参的值复制后，传送给被调用函数的形参，从而实现调用函数向被调用函数的数据传送。

函数的形参和实参具有以下特点。

1. 形参变量只有在被调用时才分配内存单元，在调用结束时，即刻释放所分配的内存单元。因此，形参只在函数内部有效，函数调用结束返回主调函数后则不能再使用该形参变量。

2. 实参可以是常量、变量、表达式、函数等，无论实参是何种类型的量，在进行函数调用时，它们都必须具有确定的值，以便把这些值传送给形参。因此应预先用赋值、输入等办法使实参获得确定值。

3. 实参和形参在数量上、类型上、顺序上应严格一致，否则会发生“类型不匹配”的错误。

4. 函数调用中发生的数据传送是单向的，即只能把实参的值传送给形参，而不能把形参的值反向传送给实参。因此在函数调用过程中，形参的值发生改变，而实参中的值不会变化。

【例题 1】求两个数之和。

```
#include "stdio.h"
void main()
{
   int a,b,c;
   scanf("%d,%d",&a,&b);
   c=add(a,b);  /* a,b 是实参*/
   printf("sum is %d\n",c);
}
int add(int x, int y)   /* x,y 是形参*/
{ int z;
  z=x+y;
  return(z);
}
```

【说明】

（1）实参可以是常量、变量、表达式、函数等。无论实参是何种类型的量，在进行函数调用时，它们都必须具有确定的值，以便把这些值传送给形参。所以我们一般在调用函数前先给实参赋值。

（2）形参变量只有在被调用时，才分配内存单元；调用结束时，即刻释放所分配的内存单元。因此，形参只有在该函数内有效。调用结束，返回调用函数后，则该形参变量失效。

（3）实参和形参占用不同的内存单元，即使同名也互不影响。

（4）实参对形参的数据传送是单向的，即只能把实参的值传送给形参，而不能把形参的值反向传送给实参。也就是说实参保留原值。

11.2.2 函数的返回值

C 语言的函数可分为有返回值函数和无返回值函数两种。

（1）有参函数的返回值，是通过函数中的 return 语句来获得的。

return 语句的一般格式：return 返回值表达式或者 return（返回值表达式）;

return 语句的功能：返回调用函数，并将“返回值表达式”的值带给调用函数。

（2）无返回值的函数

调用函数中无 return 语句，并不是不返回一个值，而是一个不确定的值。为了明确表示不返回值，可以用 void 定义成“无（空）类型”。

说明：

（1）在定义函数时，对函数类型的说明应与 return 语句中返回值表达式的类型一致。如果不一致，则以函数类型为准，即系统自动进行转换，将函数返回语句中表达式的类型转换为函数定义时的类型。

（2）如果缺省函数类型，则系统一律按整型处理。

【例题 2】求出两个数中的最大数。

```
#include "stdio.h"
void main()
{
   int a,b,c;
   int find（int a,int b）;/*说明函数*/
   scanf("%d,%d",&a,&b);
   c=find(a,b);  /*实参 a 和 b*/
   printf("max is %d\n",c);
}
int find(int x, int y)  /*形参 x 和 y*/
{ int z;
  if(x>y);
    z=x;
  else
    z=y;
  return z;
}
```

例题分析

此题中由一个主函数和一个子函数构成，在主函数中实参是 a 和 b，函数名是 find，在程序执行的时候，当从键盘上输入 a 和 b 的值后，执行 find(a,b)，调用子函数的 find(int x,int y)把 a 的值复制给 x，把 b 的值复制给 y，然后在子函数中进行 x 和 y 的选择操作，得到最大的值赋值给 z，由于 find 前面的函数类型是 int，这说明该子函数是有返回值的，

并且该返回值就是 return 后面的内容。所以，此题 z 的值就是该子函数的结果，并且把 z 的值返回给了主函数中的 c，最后输出 c 的值。

【考生注意】

关于对子函数的说明有以下几点需要注意的。

（1）子函数位于主函数下面的时候，要在主函数的定义部分对其进行说明，或者在头文件和 void main 之间进行说明。

（2）子函数位于主函数上面的时候，无须再对其进行说明。

（3）一定要把子函数的返回类型确定好，否则编写出来的程序有可能不符合原来的意思。

【例题 3】有以下程序：（2009 年 3 月）

```
#include <stdio.h>
int f(int x,int y)
{ return((y-x)*x);}
main()
{ int a=3,b=4,c=5,d;
  d=f(f(a,b),f(a,c));
  printf("%d\n",d);
}
```

程序运行后的输出结果是：

A）10　　B）9　　C）8　　D）7

例题分析

此题考查函数调用，先调用 f(3,4)，得到结果为 3，然后调用 f(3,5)，得到结果为 6，最后调用 f(3,6)，得到结果为 9，此题 B 为正确选项。

【例题 4】有以下程序：（2009 年 9 月）

```
#include <stdio.h>
void fun(int p)
{ int d=2;
  p=d++;
  printf("%d",p);
}
void main()
{ int a=1;
  fun(a);
  printf("%d\n",a);
}
```

程序运行后的输出结果是

A）32　　B）12　　C）21　　D）22

例题分析

本题主要考查主函数和子函数之间的调用关系，当 a 等于 1 的时候，调用 fun(1)执行子函数，在子函数中，p 的值经过了 d++的运算后变成了 2，由于子函数是没有返回值的，所以子函数对于主函数来说，没有任何影响，主函数中的 a 的值保持不变，结果还是 1，答案选 C。

11.2.3　函数原型的声明

调用用户自定义函数时，一般调用函数和被调用函数应在同一个文件中，在调用函数中对被调用函数返回值的类型、函数名称、函数形式参数的类型进行说明，这种说明称为函数声明。

函数原型的一般形式如下：

类型名　函数名（）；

或类型名　函数名（类型 1，类型 2，……，类型 n）；

或类型名　函数名（类型 1 形参 1，类型 2 形参 2，……，类型 n　形参 n）；

（1）函数声明是以语句形式出现的，因此其后有语句结束标记“;”。

（2）编译系统不检查参数名，因此参数名是什么都无所谓，当然也可以不写。

（3）如果被调用函数的定义出现在主函数之前，可以不用对该函数进行声明。

【例题 5】求一个数的立方和。

```
#include <stdio.h>
float cube(float  x)
{ return(x*x*x);
}
void main()
{ float a, result;
  printf("Please input value of a:");
  scanf("%f",&a);
  result=cube(a);
  printf("Cube of %.4f is %.4f\n",a,result);
}
```

（4）如果在函数的外部进行了函数声明，则在主函数中不需要再声明。如我们把例题 5 改编为如下程序。

```
#include  <stdio.h>
float cube(float x);  /*相当于在 main()函数中声明了 float cube(float x ); */
void main()
{ float a, result;
  printf("Please input value of a:");
  scanf("%f",&a);
  result=cube(a);
  printf("Cube of %.4f is %.4f\n",a,result);
}
float cube(float x)
{ return(x*x*x);
}
```

（5）除了第（3）、（4）中的情况外，我们还可以在 main 函数中声明函数原型。

例如：

```
#include <stdio.h>
void main()
{ float cube(float x );
  float a, result;
  printf("Please input value of a:");
  scanf("%f",&a);
  result=cube(a);
```

```
    printf("Cube of %.4f is %.4f\n",a,result);
}
float cube(float x)
{ return(x*x*x);
}
```

11.3　函数的嵌套调用

C 语言嵌套调用，即在被调函数中又调用其他函数。其关系可表示为如图 11-1 所示。其执行过程是：执行 void main 函数中调用 f1 函数的语句时，即转去执行 f1 函数，在 f1 函数中调用 f2 函数时，又转去执行 f2 函数，f2 函数执行完毕返回 f1 函数的断点继续执行，f1 函数执行完毕返回 void main 函数的断点继续执行。

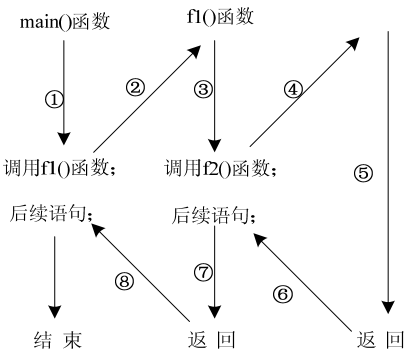


图 11-1　函数嵌套

【例题 6】求 3 个数中最大数和最小数的差值。

```
#include <stdio.h>
int dif(int x,int y,int z);
int max(int x,int y,int z);
int min(int x,int y,int z);
void main()
{  int a,b,c,d;
   scanf("%d%d%d",&a,&b,&c);
   d=dif(a,b,c);
   printf("Max-Min=%d\n",d);
}
int dif(int x,int y,int z)
{  return max(x,y,z)-min(x,y,z);
}
int max(int x,int y,int z)
{   int r;
    r=x>y?x:y;
    return(r>z?r:z);
}
```

```
int min(int x,int y,int z)
{   int r;
    r=x<y?x:y;
    return(r<z?r:z);
}
```

例题分析

（1）在定义函数时，函数 dif、max、min 是互相独立的，这从函数原型声明中可以看到。

（2）整个程序从 void main 函数开始执行，执行到 dif 函数时，出现一个 max 和 min 函数，此时先调用 max 函数，再调用 min 函数。体现出了一个被调用函数中再次调用别的函数，即为函数的嵌套调用。

11.4 函数的递归调用

在 C 语言的函数调用过程中，若函数直接或间接调用函数自身，则这种调用称为函数的“递归调用”。递归有两种类型。

（1）直接递归：函数体内直接调用函数自身。

```
void fun( )
{ ...
  fun( );
  ...
}
```

（2）间接递归：函数调用其他函数，而其他函数又调用该函数自身。

```
void fun1( )      void fun2( )
{ ...             { ...
  fun2( );          fun1( );
  ...               ...
}                    }
```

【例题 7】写出下面程序的结果。

```
int fac(int n)
{ int f;
  if(n==1)return 1;
  else
     return fac(n-1)+n;
}
void main()
{ int y;
  y=fac(5);
  printf("y =%d", y);
}
```

例题分析

首先我们要知道递归结束的条件，即 n=1。

设 n=5 时，程序执行的过程如下。

下推: f(5)=f(4)+5　　回代：　f(5)=1+2+3+4+5
　　f(4)=f(3)+4　　f(4)=1+2+3+4
　　f(3)=f(2)+3　　f(3)=1+2+3
　　f(2)=f(1)+2　　f(2)=1+2
　　f(1)=1

【例题 8】编程求裴波那契（Fibonacci）数列（1,1,2,3,5,8,…）的值。

```
#include  <stdio.h>
void main()
{ int n;
  long m;
  long fib(int);   /*函数声明*/
  scanf("%d", &n);  /*求第 n 项*/
  m=fib(n);
  printf("fib(%d)=%ld\n",n,m);
}
long fib(int n)   /*定义函数 fib 及形参说明*/
{ if（n==1）
    return（1）;
  else if(n==2)
    return (1);     /*递归终止条件*/
      else
        return fib(n-1)+fib(n-2);
}
```

例题分析

（1）对于裴波那契数列而言，有两个初始条件，分别为：

fib(1)=1，fib(2)=1；

（2）其递推关系为从第 3 项开始，每项是前两项之和。因此，可以写出它的递归公式。

fib(n)=1　(n=1 或 2)

fib(n)=fib(n-1)+fib(n-2)　(n>2)

本程序是个典型的递归调用程序，首先在主函数中调用 fib(5)，接下来执行 fib(4)+fib(3)，由于递归终止条件是 n=2，所以程序继续递归执行。同理，fib(4)调用 fib(3)+fib(2)，fib(3)调用 fib(2)+fib(1)，此时满足递归终止条件 n=2，递归结束。具体如图 11-2 所示。

数组名做函数参数时，既可以做形参，也可以做实参。数组名做函数参数时，要求形参和相对应的实参都必须是类型相同的数组，都必须有明确的数组说明。

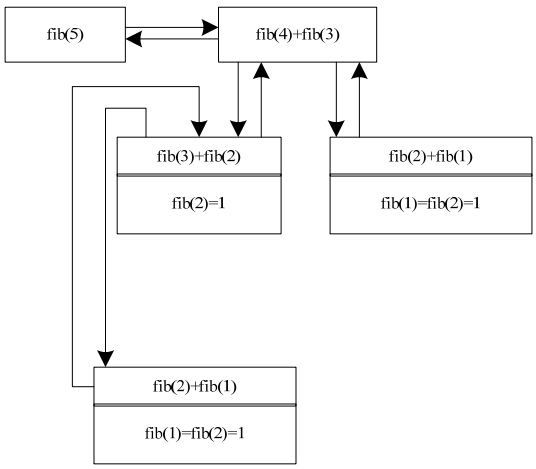


图 11-2　递归调用过程

1. 数组元素作为函数参数

数组元素就是下标变量，它与普通变量并无区别。数组元素只能用做函数实参，其用法与普通变量完全相同：在发生函数调用时，把数组元素的值传送给形参，实现单向值传送。

【例题 9】写一函数，统计字符串中数字字符的个数。

```
#include  "stdio.h"
int f(char c)
{ if  (c>='0'&&c<='9')
  return 1;
}
void main()
{ int i,num=0;
  char str[255];
  printf("Input  a  string: ");
  gets(str);
  for(i=0;str[i]!='\0';i++)
    if (f(str[i]))   num++;
  puts(str);
  printf("num=%d\n",num);
}
```

例题分析

本题在调用子函数的时候，将每一个数组元素作为参数传递给子函数，然后在子函数中对其进行判断。

2. 一维数组名可做函数参数

一维数组名做函数参数时，传递的是整个数组。此时实参与形参都应是数组名。

【例题 10】已知某个学生 6 门课程的成绩，求总成绩。

```
#include  "stdio.h"
float sum(float a[])
{ int i;
  float av,s=a[0];
  for(i=1;i<5;i++)  s+=a[i];
  return s;
}
void main()
{ float sco[6],av;
  int i;
  printf("\ninput 6 scores:\n");
  for(i=0;i<6;i++)
    scanf("%f",&sco[i]);
  av=sum(sco);
  printf(" sum is %5.2f\n",sum);
}
```

例题分析

本题主要是将数组的数组名作为函数的实参传递给形参的，将数组名作为参数传递，就是将整个数组的内容传递过去了。

【例题 11】求一个数组中最大值、最小值及平均值。

```
#include  "stdio.h"
float  max,min;
```

```
float average(float array[],int n)
{ int i; float sum=array[0];
  max=min=array[0];
  for(i=1;i<n;i++)
  { if(array[i]>max)  max=array[i];
    else if(array[i]<min)  min=array[i];
    sum+=array[i];
  }
  return(sum/n);
}
void main()
{ int i; float ave,score[20];
  scanf("%d",score);
  ave=average(score,10);
  printf("max=%6.2f\nmin=%6.2f\naverage=%6.2f\n",max,min,ave);
}
```

【例题 12】求方程 $ax^2+bx+c=0$（$a\neq0$）的实数根。

例题分析

定义函数 dict 来判断方程是否有实根，有实根则返回函数值 1，否则返回函数值 0；然后在主函数中求方程的实根。

```
#include <stdio.h>
#include <math.h>
void main()
{ float a,b,c,x1,x2,d,dt;
  int dict(float,float,float);  /*声明函数 dict 及形式参数类型*/
  printf("Input a,b,c:");
  scanf("%f,%f,%f",&a,&b,&c);/*输入方程*/
  d=dict(a,b,c);  /*调用函数 dict，传递实参 a,b,c*/
  dt=b*b-4*a*c;   /*可以用一个函数实现*/
if(d)
{ x1=(-b+sqrt(dt))/(2*a);
  x2=(-b-sqrt(dt))/(2*a);
  printf("实根 x1=%f,x2=%f\n",x1,x2);}
else
  printf("无实数根!\n");
}
int dict(float a,float b,float c)  /*定义函数 dict 及形参说明*/
{ float d;
  d=b*b-4*a*c;   /*可以用一个函数实现*/
  if(d>=0) return(1);
  else return(0);
}
```

【考生注意】

（1）用数组名做函数参数，应该在调用函数和被调用函数中分别定义数组，且数据类型必须一致，否则结果将出错。

（2）C 编译系统对形参数组大小不做检查，所以形参数组可以不指定大小。当然形参的数组也可以写为 float a[]。

（3）如果指定形参数组的大小，则实参数组的大小必须大于等于形参数组，否则因形参数组的部分元素没有确定值而导致计算结果错误。

（4）数组名做函数参数时，把实参数组的起初地址传递给形参函数，这样两个数组就共占同一段内存单元。形参数组中各元素的值发生变化，会使实参数组元素的值同时发生变化。这一点与变量函数参数的情况不同，当然也与数组元素作为函数参数不同。

3. 二维数组名做函数参数

二维数组名可以作为实际参数和形式参数，这与一维数组相同。需要注意的是我们把二维数组名当做形参时，不能省略第二维的大小说明，第一维的大小说明可有可无。

11.5 内部函数和外部函数

如果一个函数只能被本文件中其他函数所调用，它称为内部函数，内部函数又称静态函数。在定义内部函数时在函数名和函数类型前面加 static。即：

static 类型标识符 函数名（形参表）

如果一个函数可以被其他文件中的其他函数所调用，它称为外部函数。在定义函数时须冠以关键字 extern（省略也可），表示此函数是外部函数。

【例题 13】有一个字符串，内有若干个字符，今输入一个字符，程序将字符串中该字符删去，由外部函数实现。

```
file1.c(文件 1)
void main()
{ extern enter_string(),delete_string(),print_string();
  /*说明本文件要用到其他文件中的函数*/
  char c;
  static char str[80];
  enter_string(str);
  scanf("%c",&c);
  delete_string(str);
  print_string(str);
}
file2.c(文件 2)
#include "stdio.h"
extern enter_string(str) /*定义外部函数 enter_string*/
char str[80];
{ gets(str);}            /*读入字符串 str*/
file3.c(文件 3)
extern delete_string(str,ch) /*定义外部函数 delete_string*/
char str[],ch;
{ int i,j;
  for(i=j=0;str[i]!='\0';i++)
   if(str[i]!=ch)
     str[j++]=str[i];
```

```
    str[j]='\0';
}
file4.c(文件 4)
extern print_string(str)/*定义外部函数 print_string*/
char str[];
{ printf("%s",str);}
```

例题分析

整个程序由 4 个文件组成，每个文件包含一个函数。主函数是主控函数，由 4 个函数调用语句组成。其中 scanf 是库函数，另外 3 个是用户定义函数，它们都定义为外部函数。在 void main 函数中用 extern 说明在 void main 函数中用到的 enter_string()，delete_string()，print_string()是外部函数。

11.6　内部变量和外部变量

C 语言中所有的量都有自己的作用域。变量说明的方式不同，其作用域也不同。C 语言中的变量，按作用域范围可分为两种，即内部变量和外部变量。

11.6.1　内部变量

内部变量也称局部变量，是在函数内做定义说明的。其作用域仅限于函数内，离开该函数后再使用这种变量是非法的。

```
int f1(int a)       /*函数 f1*/
{
     int b,c;
     ……
}
int f2(int x)       /*函数 f2*/
{
     int y,z;
     ……
}
void main()
{
     int m,n;
     ……
  }
```

在函数 f1 内定义了 3 个变量，a 为形参，b，c 为一般变量。在 f1 的范围内 a，b，c 有效，或者说 a，b，c 变量的作用域限于 f1 内，同理，x，y，z 的作用域限于 f2 内，m，n 的作用域限于 void main 函数内。关于局部变量的作用域还要说明以下几点。

（1）主函数中定义的变量也只能在主函数中使用，不能在其他函数中使用。同时，主函数中也不能使用其他函数中定义的变量。因为主函数也是一个函数，它与其他函数是平行关系。这一点是与其他语言不同的，应予以注意。

（2）形参变量属于被调函数的局部变量，实参变量属于主调函数的局部变量。

（3）允许在不同的函数中使用相同的变量名，它们代表不同的对象，分配不同的单元，互不干扰，也不会发生混淆。如在前例中，形参和实参的变量名都为 n 是完全允许的。

（4）在复合语句中也可定义变量，其作用域只在复合语句范围内。

```
void main()
{
   int s,a;
   ……
   {
      int b;
      s=a+b;
      ……                    /*b 作用域*/
   }
   ……                   /*s,a 作用域*/
}
```

【例题 14】分析下列程序。

```
void main()
{
   int i=2,j=3,k;
   k=i+j;
   {
      int k=8;
      printf("%d\n",k);
   }
   printf("%d\n",k);
}
```

例题分析

本程序在 void main 中定义了 i，j，k3 个变量，其中 k 未赋初值。而在复合语句内又定义了一个变量 k，并赋初值为 8。应该注意这两个 k 不是同一个变量。在复合语句外由 void main 定义的 k 起作用，而在复合语句内则由在复合语句内定义的 k 起作用。因此，程序第 4 行的 k 为 void main 所定义，其值应为 5。第 7 行输出 k 值，该行在复合语句内，由复合语句内定义的 k 起作用，其初值为 8，故输出值为 8，第 9 行输出 i，k 值，i 是在整个程序中有效的，故输出也为 3。而第 9 行已在复合语句之外，输出的 k 应为 void main 所定义的 k，此 k 值由第 4 行获得为 5，故输出也为 5。

11.6.2 外部变量

外部变量也称全局变量，它是在函数外部定义的变量，它不属于哪一个函数，它属于一个源程序文件，其作用域是整个源程序。在函数中使用全局变量，一般应做全局变量说明，只有在函数内经过说明的全局变量才能使用。全局变量的说明符为 extern。但在一个函数之前定义的全局变量，在该函数内使用可不再加以说明。

```
int a,b;          /*外部变量*/
void f1()         /*函数 f1*/
{
   ……
}
float x,y;        /*外部变量*/
int fz()          /*函数 fz*/
```

```
{
   ……
}
void main()          /*主函数*/
{
  ……
    }
```

说明：

从上例可以看出 a，b，x，y 都是在函数外部定义的外部变量，都是全局变量。但 x，y 定义在函数 f1 之后，而在 f1 内又无对 x，y 的说明，所以它们在 f1 内无效。a，b 定义在源程序最前面，因此在 f1，f2 及 void main 内不加说明也可使用。

【例题 15】分析下面程序。

```
int a=3,b=5;     /*a，b 为外部变量*/
max(int a,int b) /*a，b 为外部变量*/
{ int c;
  c=a>b?a:b;
  return(c);
}
void main()
{ int a=8;
  printf("%d\n",max(a,b));
}
```

例题分析

如果同一个源文件中，外部变量与局部变量同名，则在局部变量的作用范围内，外部变量被“屏蔽”，即不起作用。

11.7　变量的动态存储和静态存储

在 C 语言中，对变量的存储类型说明有以下 4 种：自动变量（auto）、寄存器变量（register）、外部变量（extern）、静态变量（static）。自动变量和寄存器变量属于动态存储方式，外部变量和静态变量属于静态存储方式。

静态存储是在程序运行期间分配固定的存储空间，动态存储是在程序运行期间分配动态的存储空间。

全局变量与局部静态变量（static）存储在静态存储空间，是在编译时分配在静态存储区的，编译的时候初始化。static 全局变量（静态外部变量）的目的是让其他文件不能用 extern 引用这个文件中的全局变量。

局部静态变量说明。

1. 局部静态变量存储在静态存储单元。
2. 静态变量是在编译的时候赋初值的，只赋初值一次。普通变量是在程序运行时赋初值的。
3. 若静态变量不赋初值，在编译的时候会自动赋初值 0 或空字符。普通变量不赋初值会得到一个不确定的值。
4. 只有定义全局变量和局部静态变量的时候才能对数组初始化。动态存储区内的数组不能初始化，但是可以赋值。
5. 局部静态变量在函数调用完仍然存在，但是其他函数不能引用。

局部变量的值放在运算器的寄存器中，提高效率，叫做寄存器变量，register。只有局部自动变量和形式参数可以作为寄存器变量，全局变量与局部静态变量不能放在寄存器中。机器不同寄存器变量容许的个数不同。MS C，Turbo C 对 register 变量当成自动变量处理，不会提高效率。

全局变量的引用方式：同文件中，如果函数想引用其后定义的全局变量，需要用 extern 先对变量进行说明，然后才能引用；如果是不同文件中的引用，需要在其他文件的开头用 extern 进行变量说明。如果在一个文件中的全局变量前加 static，那就是静态外部变量，其他文件不能用 extern 引用它。

11.8 习题

11.8.1 选择题

1. 下列程序的输出结果是（　　）。

```
#include <stdio.h>
void f(int *x,int *y)
{ int t;
  t=*x,*x=*y;*y=t;
}
main()
{ int a[8]={1,2,3,4,5,6,7,8},i,*p,*q;
  p=a;q=&a[7];
  while(p<q)
  { f(p,q);p++;q--;}
    for(i=0;i<8;i++)
       printf("%d,",a[i]);
}
```

A）8,2,3,4,5,6,7,1　　B）5,6,7,8,1,2,3,4　　C）1,2,3,4,5,6,7,8　　D）8,7,6,5,4,3,2,1

2. 设函数 fun 的定义形式为：

```
void fun(char ch,float x) {…}
```

则以下对函数 fun 的调用语句中，正确的是（　　）。

A）fun("abc",3.0);　　B）t=run('D',16.5);　　C）fun('65',2.8);　　D）fun(32,32)

3. 有以下程序：

```
#include <stdio.h>
int fun1(double a){return a*=a;}
int fun2(double x,double y)
{ double a=0,b=0;
  a=fun1(x);b=fun1(y);return(int)(a+b);
}
main()
{double w;w=fun2(1.1,2.0);printf("%f",w)}
```

程序执行后变量 w 中的值是（　　）。

A）5.21　　B）5　　C）5.0　　D）0.0

4. 有以下程序：

```
#include <stdio.h>
int fun(int x,int y){return(x+y);}
main()
{ int a=1,b=2,c=3,sum;
  sum=fun((a++,b++,a+b),c++);
  printf("%d\n",sum);
}
```

程序执行后的输出结果是（　　）。

A）6　　B） 7　　C）8　　D）9

5. 有以下程序：

```
#include <stdio.h>
int fun(int x)
{
   int p;
   if(x==0||x==1)return(3);
   p=x-fun(x-2);
   return p;
}
main()
{printf("%d\n",fun(7));}
```

程序执行后的输出结果是（　　）。

A）7　　B）3　　C）2　　D）0

6. 调用函数时，当实参和形参都是简单变量时，它们之间的数据传递过程是（　　）。

A）实参将其值传递给形参，调用结束时形参再将其值回传给实参

B）实参将其地址传递给形参，调用结束时形参再将其地址回传给实参

C）实参将其地址传递给形参，并释放原先占用的存储单元

D）实参将其值传递给形参，调用结束时形参并不将其值回传给实参

7. 若要计算 tan(60)的值，则调用的库函数格式为（　　）。

A）tan(60)　　B）tan(pi/3)　　C）tan(60.0)　　D）tan((double)60)

8. 函数调用语句“func((exp1,exp2),(exp3,exp4,exp5));”中含有实参的个数为（　　）。

A）1　　B）2　　C）3　　D）5

9. 下列程序的输出结果是（　　）。

```
#include <stdio.h>
void p(int *x)
{ printf("%d",++*x);
}
void main()
{ int y=3;
  p(&y);
}
```

A）3　　B）4　　C）2　　D）5

10. C 语言规定，程序中各函数之间（　　）。

A）既允许直接递归调用也允许间接递归调用

B）不允许直接递归调用也不允许间接递归调用

C）允许直接递归调用不允许间接递归调用

D）不允许直接递归调用允许间接递归调用

11. 有以下程序：

```
#include <stdio.h>
int fun(int x,int y){return(x+y);}
main()
{ int a=1,b=2,c=2,sum;
  sum=fun((a++,b++,a+b),c++);
  printf("%d\n",sum);
}
```

执行后的结果是（　　）。

A）5　　B）6　　C）7　　D）8

12. 下列程序的输出结果是（　　）。

```
#include <stdin.h>
#include <stdio.h>
void fun(int b[])
{  static int i=0;
   do
   { b[i]+=b[i+1];
   }while(++i<2);
}
main()
{ int k,a[5]={1,3,5,4,9};
  fun(a);
  for(k=0;k<5;k++)printf("%d",a[k]);
}
```

A）13579　　B）48579　　C）48549　　D）48999

13. 有以下程序：

```
#include <stdio.h>
int fun(int x)
{ int p;
  if(x= =0 || x= =1)return(3);
  p=x-fun(x-2);
  return p;
}
main()
{printf("%d\n",fun(7));}
```

程序执行后的输出结果是（　　）。

A）7　　B）3　　C）2　　D）0

14. 以下程序的输出结果是（　　）。

```
#include <studio.h>
int f(int a)
{ return a%2;}
main()
{ int s[8]={1,3,5,2,4,6},i,d=0;
  for(i=0;f(s[i]);i++)d+=s[i];
  printf("%d\n",d);
}
```

A）9　　B）11　　C）19　　D）21

15.有以下程序：

```
#include <stdio.h>
void change(int k[]){k[0]=k[5];}
main()
{int x[10]={1,2,3,4,5,6,7,8,9,10},n=0;
 while(n<=4){change(&x[n]);n++;}
 for(n=0;n<5;n++)printf("%d",x[n]);
 printf("\n");
}
```

程序运行后的输出结果是（　　）。

A）6 7 8 9 10　　B）1 3 5 7 9　　C）1 2 3 4 5　　D）6 2 3 4 5

16. 有以下程序：

```
#include <stdio.h>
void fun2(char a,char b )   {printf("%c%c",a,b);}
char a='A',b='B';
void fun1(){a='C';b='D';}
main()
{ fun1();
  printf("%c%c",a,b);
  fun2('E','F');
}
```

程序的运行结果是（　　）。

A）CDEF　　B）ABEF　　C）ABCD　　D）CDAB

17. 有以下程序：

```
#include <stdio.h>
void f(int b[])
{int i;
 for(i=2;i<6;i++)
 b[i]*=2;
```

```
}
main()
{int a[10]={1,2,3,4,5,6,7,8,9,10},i;
 f(a);
 for(i=0;i<10;i++)
   printf("%d,",a[i]);
}
```

程序运行后的输出结果是（　　）。

A）1,2,3,4,5,6,7,8,9　　B）1,2,6,8,10,12,7,8,9,10

C）1,2,3,4,10,12,14,16,9,10　　D）1,2,6,8,10,12,14,16,9,10

11.8.2　填空题

1. 下列程序运行后的输出结果是_____。

```
void swap(int x,int y)
{ int t;
  t=x;x=y;y=t;printf("%d %d",x,y);
}
main()
{
  int a=3,b=4;
  swap(a,b);printf("%d %d\n",a,b);
}
```

2. 下列程序运行后的输出结果是_____。

```
int f(int a[],int n)
{ if(n>=1)return f(a,n-1)+a[n-1];
  else return  0;
}
main()
{ int aa[5]={1,2,3,4,5},s;
  s=f(aa,5); printf("%d\n",s);
}
```

3. 下列程序的输出结果是_____。

```
#include <stdio.h>
int sb(int s[],int b)
{ static int n=3;
  b=s[n];
  n--;
  return(b);
}
main()
```

```
{ int s[]={1,5,6,8};
  int i,x=0;
  for (i=0;i<4;i++)
  { x=sb(s,x);
    printf("%d",x);
  }
  printf("\n");
}
```

4. 下面程序的功能是：将 N 行 N 列二维数组中每一行的元素进行排序，第 0 行从小到大排序，第 1 行从大到小排序，第 2 行从小到大排序，第 3 行从大到小排序，例如：

当　A=2　3　4　1
　　8　6　5　7
　　11　12　10　9
　　15　14　16　13

则排序后 A= 1　2　3　4
　　8　7　6　5
　　9　10　11　12
　　16　15　14　13

请在程序空白处填入适当内容，使程序完整。

```
#include <stdio.h>
void sort(int a[][4])
{  int i,j,k,t;
   for(i=0;i<4;i++)
   for(j=0;j<3;j++)
   for(k=____;k<N;k++)
   /*判断行下标是否为偶数来确定按升序或降序来排序*/
     if(_____?a[i][j]<a[i][k]:a[i][j]>a[i][k])
     {
        t=a[i][k];
        a[i][k]=a[i][j];
        a[i][j]=t;
   }
}
void outarr(int a[4][4])
{……}
main()
{ int aa[4][4]={{2,3,4,1},{8,6,5,7},{11,12,10,9},{15,14,16,13}};
  outarr(aa);/*以矩阵的形式输出二维数组*/
  sort(aa);
  outarr(aa);
}
```

5. 下列程序运行后的输出结果是_____。

```
#include <stdio.h>
int fun(int a)
{
   int b=0;static int c=3;
   b++;c++;
   return(a+b+c);
}
main()
{
   int i,a=5;
   for(i=0;i<3;i++)printf("%d%d",i,fun(a));
   printf("\n");
}
```

6. 有以下程序:

```
#include <stdio.h>
int sub(int n) {return(n/10+n%10);}
main()
{  int x,y;
   scanf("%d",&x);
   y=sub(sub(sub(x)));
   printf("%d\n",y);
}
```

若运行时输入：1234<回车>，则程序的输出结果是_____。

11.8.3 程序设计题

1. 用冒泡法把一组数组从大到小进行排序。
2. 编程求两个整数的阶乘之和。

第12章 指针

本章主要介绍指针与指针变量的概念，指针与数组的关系及指针与函数的关系，根据计算机等级考试的要求，具体如表 12-1 所示。

表 12-1　考试要求

考试知识点	重要性
指针的概念	★★
指针变量的概念	★★
指针变量的定义和使用	★★★
指针与数组	★★★★★
指针与函数	★★★★★

12.1　指针的概念

1. 变量的地址

在讲指针的概念之前，我们先来理解关于计算机内存中地址的概念，打个比方：假设在高中有高二（3）这么一个班，这个班在这所高中就要有个教室，那么这个教室也就有一个门牌号，而这个门牌号也就是班级所在的教室的称呼（3 号楼 103），那么将整个计算机的内存看成一所学校，把高二（3）班看成是一个变量，那么 3 号楼 103 就是高二（3）班（变量）在学校（计算机内存）中的地址。例如 int m，当计算机编译这条定义变量语句的时候，其实就是给这个变量分配了内存单元。计算机在内存中开辟一个 2 个字节的空间来存储这个整型变量 m 的内容（假设存储单元为 FF34，FF35）。而在内存中开辟空间的那个起始位置（即 FF34）就是变量 m 在内存中的地址。

2. 指针的定义

通过以上例子讲解了内存变量的地址和内存变量的值，那么指针是什么呢？

一个变量的地址就是该变量的指针。结合上面的例子，变量 i 的地址就是变量 i 的指针，也就是说变量 i 的指针是 FF34。

3. 使用指针的目的

有的时候在进行函数调用的时候及后面要讲的数据结构中，如果函数参数传递的是变量的地址的时候，就会使用到指针作为参数进行传递，另外，在以后的数据结构的学习中，也会遇到指针这样的概念，总之指针的使用比较广泛。

12.2 指针变量的定义和类型

12.2.1 指针变量的定义

指针变量定义的一般形式为：

类型标识符 *标识符

对指针变量的类型说明包括 3 个内容：

（1）指针类型说明，即定义变量为一个指针变量。

（2）指针变量名。

（3）变量值（指针）所指向的变量的数据类型。

就如同变量有 3 种类型一样，指针变量也有 3 种类型：整型指针变量、浮点数型指针变量、字符型指针变量。

例如：

```
int *p1;
float *p1;
char *p1;
```

说明：

（1）标识符前面的*，表示该变量为指针变量，指针变量名是 p1。

（2）一个指针变量只能指向同一类型的变量，也就是说只能将变量的地址赋值给 p1。

12.2.2 指针变量的运算

（1）赋值运算

指针变量的赋值运算有以下几种形式。

①指针变量初始化赋值，前面已做介绍。

②把一个变量的地址赋给指向相同数据类型的指针变量。

例如：

```
int a,*pa;
pa=&a; /*把整型变量 a 的地址赋给整型指针变量 pa*/
```

③把一个指针变量的值赋给指向相同类型变量的另一个指针变量。

例如：

```
int a,*pa=&a,*pb;
pb=pa; /*把 a 的地址赋给指针变量 pb*/
```

由于 pa，pb 均为指向整型变量的指针变量，因此可以相互赋值。

④把数组的首地址赋给指向数组的指针变量。

例如：int a[5],*pa;

```
pa=a; /*数组名表示数组的首地址，故可赋给指向数组的指针变量 pa*/
```

也可写为：

```
pa=&a[0]; /*数组第一个元素的地址也是整个数组的首地址，也可赋给 pa*/
```

当然也可采取初始化赋值的方法：

```
int a[5],*pa=a;
```

⑤把字符串的首地址赋给指向字符类型的指针变量。例如：“char *p;p="c language";”可用初始化赋值的方法写为“char *p="C Language";”这里应说明的是并不是把整个字符串装入指针变量，而是把存放该字符串的字符数组的首地址装入指针变量。

12.2.3　指针变量的引用

【例题 1】指针变量的引用。

```
#include  "stdio.h"
 void main()
{  int m=5;            /*开辟空间来存储 m 的值，空间的地址是 FF4A*/
   int *p;                  /*定义整型指针变量*/
   p=&m;               /*将 m 的地址（假设 FF4A）赋值给指针 p*/
   printf("%d",*p);         /*输出的值为 3*/
}
```

例题分析

（1）第 1 个*p 表示定义了一个指针变量 p，用来存储整型变量的地址。

（2）第 2 个*p 表示输出指针 p 的值在内存中对应的单元内容。

【例题 2】指针变量的引用。

```
#include  "stdio.h"
void main()
{  int m=3;            /*开辟空间来存储 m 的值，空间的地址是 FF4A*/
   int *p;                  /*定义整型指针变量*/
   p=&m;               /*将 m 的地址（假设 FF4A）赋值给指针 p*/
   printf("%x",p);          /*输出指针 p 的值*/
   printf("%d",m);          /*输出的值为 3*/
   printf("%d",*p);         /*输出的值为 3*/
}
```

指针变量的引用如图 12-1 所示。

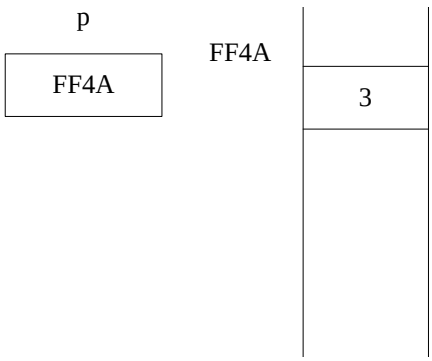


图 12-1　指针变量的引用

【考生注意】

（1）在本程序中出现了两个*p，第 1 个*p 表示定义了一个指针变量 p，用来存储整型变量的地址，第 2 个*p 表示输出指针 p 的值在内存中对应的单元内容，也就是指针 p 的值（FF4A）在内存中所对应的单元的值（即 m 的值为 3）。

（2）“printf("%x",p);”表示输出指针 p 的值，由于内存中是以十六进制方式存储的，所以输出的格式为 x，本句主要是想让大家对指针有个直观的认识。

【例题 3】输入两个整数，按从小到大的顺序输出。

```
#include  "stdio.h"
void main()
{ int a,b,*p,*q,*t;
  scanf("%d%d",&a,&b);
  p=&a;       /*将变量 a 的地址赋值给指针 p*/
  q=&b;       /*将变量 b 的地址赋值给指针 q*/
  if(*p>*q)  /*如果变量 a 的值大于变量 b 的值，进行交换*/
  { t=p;      /*将指针 p 的值赋值给 t*/
    p=q;      /*将指针 q 的值赋值给 p*/
    q=t;      /*将指针 t 的值赋值给 q*/
  }
  printf("结果是：%d %d",*p,*q);
}
```

例题分析

交换结果如图 12-2 所示。

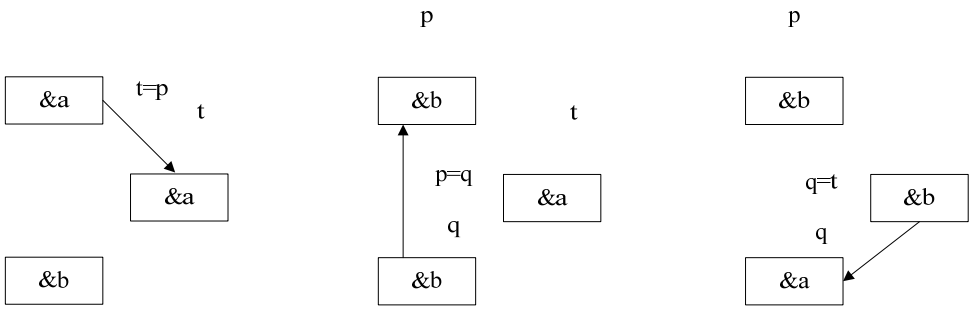


图 12-2　两个数的交换

【例题 4】有以下程序：（2009 年 9 月）

```
#include  <stdio.h>
void main()
{ int m=1,n=2,*p=&m,*q=&n,*r;
  r=p;p=q;q=r;
  printf("%d,%d,%d,%d\n",m,n,*p,*q);
}
```

程序运行后的输出结果是（　　）。

A）1,2,1,2　　B）1,2,2,1　　C）2,1,2,1　　D）2,1,1,2

例题分析

指针的使用，指针内容交换，涉及变量的值是否进行交换，是考试的一个方面。在本题中，指针 p 的值等于 m 的地址，指针 q 的值等于 n 的地址，然后将指针 p 和指针 q 进行交换，也就是说指针 p 的值变成了 n 的地址，指针 q 的值变成了 m 的地址，但是 m 和 n 的内容并没有发生改变，还是原来的值，所以，此题的答案是 1,2,2,1，选 B。

12.3　指针与一维数组

在前面讲述了关于一维数组的使用，而在实际的 C 语言使用过程中，是将指针与一维数组结合起来使用的，将一维数组的使用引入了更广泛的空间。

12.3.1　一维数组指针的定义

一维数组指针是指数组在内存中的起始地址，即第一个数组元素在内存中的地址。

格式如下：

数据类型 数组名[长度]，指针变量；

指针变量＝数组名

或者

数据类型 数组名[长度]；

数据类型 指针变量；

指针变量＝数组名

12.3.2　一维数组指针的使用

```
#include  "stdio.h"
void main()
{ int a[5],i;              /*定义一维整型数组，数组名为 a*/
  int *p;                       /*定义了一个整型指针变量 p*/
  p=a;                     /*把数组名 a 赋值给指针变量 p，即指针 p 指向第一个元素的地址*/
  for(i=0;i<5;i++)
    { printf("%d",a[i]);
      printf("%d",*(p+i));      /*p+i 指向第 i 个的地址，*(p+i)表示第 i 个元素的值*/
      printf("%d",*(a+i));      /*a+i 指向第 i 个的地址，*(a+i)表示第 i 个元素的值*/
 }
```

说明：

（1）数组名有两个含义，一是表示对一个数组的称呼，二是表示该数组在内存中的地址。

（2）“int *p=a;”包含了两条语句，第一条语句表示定义了一个整型指针变量 p，第二条语句表示将数组名赋值给指针 p（p=a），即将数组在内存中的地址赋值给指针 p。

（3）p+i，p-i 表示将指针从当前位置向前（+i）或回退（-i）i 个数据单位，而不是简单的地址加 1，而是指向下一个变量的地址。

【例题 5】有以下程序：（2009 年 9 月）

```
#include <stdio.h>
#include <string.h>
void main()
{ char str[ ][20]={"One*World", "One*Dream"},*p=str[1];
  printf("%d,",strlen(p));printf("%s\n",p);
}
```

程序运行后的输出结果是（　　）。

A）9,One*World　　B）9,One*Dream　　C）10,One*Dream　　D）10,One*World

例题分析

本题主要是考查二维数组与指针的用法，题目定义了一个二维数组，一个指针 p，而 p 的值是 str[1]的首地址，也就是说指针 p 指向“One*Dream”，所以，strlen(p)为 9，答案选 B。

12.4 指针与二维数组

二维数组是在一维数组的基础上进行的扩充，而指针与二维数组的使用可以更好地了解二维数组中数的存储。

12.4.1 二维数组指针的定义

1. 二维数组指针的定义

在 C 语言中定义的二维数组实际上是一个一维数组，这个一维数组的每个元素又是一个一维数组。二维数组指针是指二维数组在内存中的起始地址，即第一个数组元素在内存中的地址。

二维数组是在一维数组的基础上按照长度 2 对每个一维数组的长度 1 进行等分。那么二维数组的指针在使用的时候也是从一维数组中引入的。

格式如下：

数据类型 数组名[长度 1][长度 2]，（*指针变量）[长度 2]；

指针变量＝数组名

例如：

int a[3][3]

可见 a 数组由 a[0]、a[1]、a[2]3 个元素组成。可用 a[0][0]、a[0][1]等来引用，其他以此类推。

C 语言中，在函数体中或在函数外部定义的一维数组名是一个地址常量，其值为数组第一个元素的地址，此地址的基类型就是数组元素的类型。在以上的二维数组中，a[0]、a[1]、a[2]都是一维数组名，同样也代表一个不可变的地址常量，其值依次为二维数组每行第一个元素的地址，其基类型就是数组元素的类型。

2. 二维数组名也是一个地址值常量

二维数组名同样也是一个存放地址常量的指针，其值为二维数组中第一个元素的地址。以上 a 数组，数组名 a 的值与 a[0]的值相同，只是其基类型为具有 4 个元素的数组类型，即 a+0 的值与 a[0]的值相同，a+1 的值与 a[1]的值相同，a+2 的值与 a[2]的值相同，它们分别表示 a 数组中第 1、第 2、第 3 行的首地址。二维数组名应理解为一个行指针。

3. 二维数组元素的地址

二维数组元素的地址可以由表达式&a[i][j]求得，也可以通过每行的首地址来表示。以上二维数组 a 中，每个元素的地址可以通过每行的首地址 a[0]，a[1]，a[2]等来表示。如地址&a[0][0]可以用 a[0]+0 来表示，地址&a[0][1]可以用 a[0]+1 来表示，则 a[i][j]的地址可用以下 5 种表达式求得。

（1）&a[i][j]。

（2）a[i]+j。

（3）*(a+i)+j。

（4）&a[0][0]+3*i+j　　/*在 i 行前尚有 4*i 个元素存在*/。

（5）a[0]+3*i+j。

【例题 6】二维数组指针的定义。

```
#include "stdio.h"
void main()
{ int a[3][2]={{1,2},{3,4},{5,6}},(*p)[2],i,j;/*定义了指针 p*/
  p=a;                                  /*将数组名赋值给指针变量 p*/
```

```
    for(i=0;i<3;i++)
    for(j=0;j<2;j++)
    printf("%3d",*(*(p+i)+j);    /*通过指针 p 的移动求出相应的数组元素的值*/
}
```

【考生注意】

（1）p+i：表示 a[i-1]的地址。

（2）*(p+i)：表示 a[i-1]的值，但请注意，在二维数组中，a[i-1]又是数组元素 a[i-1][0]和 a[i-1][1]两个数组元素的数组名，所以表示的仍然是地址。

（3）*(p+i)+j：表示 a[i-1][j]元素的地址。

（4）*(*(p+i)+j)：表示 a[i-1][j]元素的值。

二维数组指针表示如图 12-3 所示。

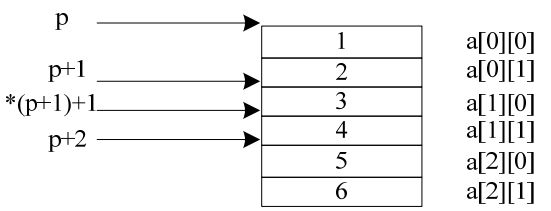


图 12-3　二维数组指针表示

12.4.2　二维数组指针的理解

假设二维数组 a[3][2]={{1,2},{3,4},{5,6}};其中 a 是数组名（假设每一个数组元素占据 2 个字节），数组中包含了 a[0]，a[1]，a[2]，而 a[0]，a[1]，a[2]又是一维数组，每个包含了 2 个元素，即上图中 a[0][0]，a[0][1]，a[1][0]，a[1][1]，a[2][0]，a[2][1]。作为数组名 a 表示整个二维数组的首地址（假设这个二维数组的首地址是 2000），也就是 a[0]的地址，而 a+1 表示 a[1]的地址，其值是 2004，同理 a+2 表示 a[2]的地址，其值是 2008，由于 a[0]，a[1]，a[2]又包含了 2 个元素，所以 a[0][0]的地址即 2000，a[0][1]的地址是 a[0]+1 即 2002，a[1][0]的地址即是 a[1]的地址即 2004，a[1][1]的地址是 a[1]+1 即 2006，a[2][0]的地址即是 a[2]的地址即 2008，a[2][1]的地址是 a[2]+1 即 2010，如图 12-4 所示。

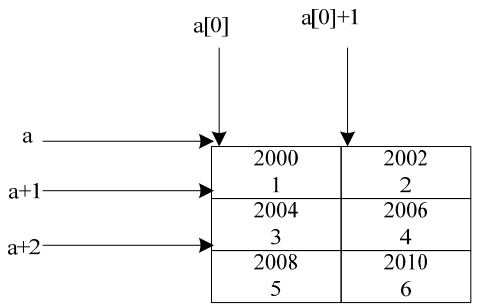


图 12-4　二维数组的理解

从上可以看出，a[0]和*(a+0)是等价的，a[1]和*(a+1)是等价的，a[2]和*(a+2)是等价的。因此 a[0]+1 和*(a+0)+1 的值都是&a[0][1]，想进一步得到 a[0][1]的值，那么就必须用如下的方式*(a[0]+1)，*(*(a+0)+1)来表示了。

最后对 a[i]的性质做进一步分析，a[i]从形式上看是 a 数组中的第 i 个元素，如果 a 表示的是一个一维数组名，则 a[i]表示 a 数组中第 i 个元素所占的内存单元。a[i]是有物理地址的，是占用内存空间的，但是如果 a 表示的是二维数组，则 a[i]代表一维数组名，a[i]本身并不占据内存单元，它也不存放 a 数组中的各个元素的值，它只是一个地址，a，a+i，a[i] *(a+i)，*(a+i)+j，a[i]+j 表示的都是地址。*(a[i]+j)，*(*(a+i)+j)是二维数组元素 a[i][j]的值，如表 12-2 所示。

表 12-2　二维数组的表示方式

表示方式	含义	地址
a	二维数组名，数组首地址	2000
a[0],*(a+0),*a	第 0 行第 0 列的首地址	2000
a+1	第 1 行的首地址	2004
a[1],*(a+1)	第 1 行第 0 列元素的地址	2004
a[1]+1,*(a+1)+1,&a[1][1]	第 1 行第 1 列元素的地址	2008
(a[1]+2),(*(a+1)+2),a[1][2]	第 1 行第 1 列元素的值	6

12.4.3 通过地址引用二维数组元素

若有以下定义：

int a[3][3],i,j;

0=<i<=3，0=<j<=4，则 a 数组元素可用以下 5 种表达式来引用。

（1）a[i][j]。

（2）* a[i]+j。

（3）*(*(a+i)+j)。

（4）(* (a+i))[j]。

（5）*(&a[0][0]+4*i+j)。

12.4.4 通过建立一个指针数组引用二维数组元素

若有以下定义：

int *p[3],a[3][2],i,j ;

在这里，说明符*p[3]中也遵照运算符的优先级，一对[]的优先级高于*号，因此 p 首先与[]结合，构成 p[3]，说明了 p 是一个数组名，系统将为它开辟 3 个连续的存储单元；在它前面的*号则说明了数组 p 是指针型的，它的每个元素都是类型为 int 的指针。若满足条件：0<=i<=3，则 p[i]和 a[i]的基类型相同，p[i]=a[i]是合法的赋值表达式。若有以下循环：

for（i=0;i<3;i++）p[i]=a[i];

在这里，等号右边的 a[i]是常量，表示 a 数组每行的首地址，等号左边的 p[i]是指针变量，循环执行的结果是 p[0]、p[1]、p[2]分别指向 a 数组每行的开头。它们的等价形式如下。

（1）*(p[i]+j)　　/*与*(a[j]+j)对应*/。

（2）*(*(p+i)+j)　　/*与*(*(a+i)+j)对应*/。

（3）(*(p+i))[j])　　/与(*(a+i))[j]对应*/。

（4）p[i][j]　　/*与 a[i][j]对应*/。

不同的是：p[i]中的值是可变的，而 a[i]中的值是不可变的。

指针数组 p[3]的表示如图 12-5 所示。

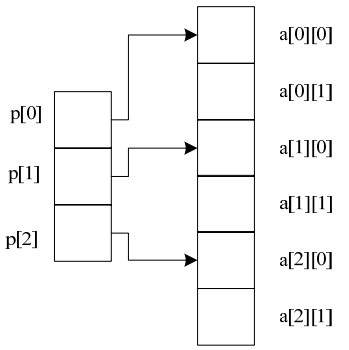


图 12-5　指针数组 p[3]的表示

12.4.5 通过建立一个行指针引用二维数组元素

若有以下定义：

int a[3][2],(*p)[2];

在说明符(*p)[2]中，由于一对圆括号的存在，所以*号首先与 p 结合，说明 p 是一个指针变量，然后再与说明符[2]结合，说明指针变量 p 的基类型是一个包含有两个 int 元素的数组。在这里，p 的基类型与 a 是相同的，因此“p=a;”是合法的赋值语句。p+1 等价于 a+1，等价于 a[1]。当 p 指向 a 数组的开头时，可以通过以下形式来引用 a[i][j]。

（1）*(p[i]+j)　　/*与*(a[i]+j)对应*/。

（2）*(*(p+i)+j)　　/*与*(*(a+i)+j)对应*/。

（3）(*(p+i))[j]　　/*与(*(a+i))+[j]对应*/。

（4）p[i][j]　　/*与 a[i][j]对应*/。

在这里，p 是指针变量，它的值可变，而 a 是一个常量。

12.4.6 二维数组指针的使用

【例题 7】求出下列程序的结果。

```
#include "stdio.h"
```

```
void main()
{ int a[3][3]={1,3,5,6,7,8,9,11,5},(*p)[3];
  p=a;
  int i,j;
  int sum=0;
  for(i=0;i<3;i++)
  { for(j=0;j<3;j++)
      if(i==j)            /*此条件用来求出主对角线元素之和*/
        sum=sum+*(*(p+i)+j);
      if(i+j==2)      /*用来求出另一条对角线元素之和*/
        sum=sum+*(*(p+i)+j);
  }
  printf("sum=%d",sum);        //结果为 40
}
```

【例题 8】求出下列程序的结果。

```
#include  "stdio.h"
void main()
{ int a[3][4]={8,4,2,11,7,8,9,3,1,23,34,11},(*p)[4];
  int i,j,max;
  for(i=0;i<4;i++)
  { max=*(*(p+i)+0);         /*假设第 i 行的第 0 个元素的值最大*/
    for(j=0;j<4;j++)
       if(max<*(*(p+i)+j))       /*如果最大值还小于该行中的某个元素，就进行重新赋值*/
          max=*(*(p+i)+j);
    printf("%d\n",max);       /*输出每一行的最大值*/
  }
}
```

12.4.7　字符串指针的定义

字符串在内存的起始地址称为字符串指针，可以定义一个字符指针变量指向一个字符串。

格式如下：

字符类型　指针变量＝字符串；

或者

字符类型 数组名[长度]；

字符类型 指针变量；

指针变量＝字符串；

【例题 9】字符串指针的定义。

```
#include "stdio.h"
void main()
{ char *string="I Love china";       /*定义指针变量 string 指向字符串*/
  printf("%s\n",string);             /*输出直接使用指针来表示*/
}
```

【考生注意】

（1）在这里没有定义字符数组，但 C 语言对字符串常量是按照字符数组来处理的，实际上在内存中开辟了一个字符数组用来存放该字符串常量。在该程序中定义了一个指针变量 string，并把该字符串首地址赋值给了 string。%s 表示输出的是一个字符串，系统先输出字符串的第 1 个字符，然后自动将 string 加 1，接着输出第 2 个字符，直到字符串的结束标志‘\0’为止。

（2）通过字符数组名或者指向字符串的指针变量，可以将字符数组中各元素的值一次性输出，而不需要借助循环语句输出，这与其他类型的数组是不同的。

【例题 10】将字符串 b 接入到字符串 a。

```
#include  "stdio.h"
#include "string.h"
void main()
{ char *a="I love Beijing";
  char *b="and I love Changsha";
  strcat(a,b);     /*使用连接函数将两个字符串连接起来*/
  puts(a);         /*结果为：I love beijing and I love shanghai*/
}
```

【例题 11】将字符串中的大写字母改写成小写字母。

```
#include  "stdio.h"
void main()
{ char *a="AerYRacV";
  int i;
  for(i=0;*(a+i)!='\0';i++)
    {if(*(a+i)>='a'&&*(a+i)<='z')
       *(a+i)=*(a+i)+32;
     printf("%c",*(a+i));/*结果为：AERYRACV*/
    }
}
```

【例题 12】以下选项中正确的语句组是（　　）。（2009 年 3 月）

A）char　s[ ];s="BOOK!";　　B）char　*s;s={"BOOK!"};

C）char　s[10];s="BOOK!";　　D）char　*s;s="BOOK!";

例题分析

A 中数组名不能直接赋值，B 中不需要{} ，C 和 A 一样不能数组名直接赋值。所以选 D。

【例题 13】设有定义“char *c;”，以下选项中能够使字符型指针 c 正确指向一个字符串的是（　　）。（2009 年 9 月份）

A）char str[ ]="string";c=str;　　B）scanf("%s",c);

C）c=getchar();　　D）*c="string";

例题分析

此题主要考查字符型指针变量的定义，由定义，得到此题选 A。

12.4.8　使用字符串指针变量与字符数组的区别

用字符数组和字符指针变量都可实现字符串的存储和运算，但是两者是有区别的。在使用时应注意以下几个问题。

1. 字符串指针变量本身是一个变量，用于存放字符串的首地址。而字符串本身存放在以该首地址为首的一块连续的内存空间中，并以‘\0’作为字符串的结束。字符数组是由若干个数组元素组成的，它可用来存放整个字符串。

2. 对字符数组做初始化赋值，必须采用外部类型或静态类型，如“s char st[]={"C Language"};”，而对字符串指针变

量则无此限制，如“char *ps="C Language";”。

3. 对字符串指针方式“char *ps="C Language";”可以写为“char *ps;　ps="C Language";”。而对数组方式

“static char st[]={"C Language"};”

不能写为

“char st[20];st={"C Language"};”，而只能对字符数组的各元素逐个赋值。

从以上几点可以看出字符串指针变量与字符数组在使用时的区别，同时也可看出使用指针变量更加方便。前面说过，当一个指针变量在未取得确定地址前使用是危险的，容易引起错误，但是对指针变量直接赋值是可以的。因为 C 系统对指针变量赋值时要给以确定的地址。

12.5　指针与函数

在函数调用中，我们经常使用指针来作为参数进行传递，使用指针作为函数的参数可以使程序的调用变得更加灵活和方便。

12.5.1　指针数组的定义

在前面学习的整型数组、字符数组等都是用来存储相对应类型的常量的，那么指针数组是用存储什么的呢？所谓指针数组，就是指数组元素都是指针类型数据的数组，也就是说指针数组中的每一个元素都是指针变量。

格式如下：

类型标识 *数组名[数组长度]

【例题 14】指针数组的定义。

```
#include  "string.h"
#include  "stdio.h"
void main()
{  char *nation[3]={"beijing","hanghzou","nanjing"};/*定义指针数组 p[3]，p[0]存储第 1 个字符串的首地
   址，p[1]存储第 2 个字符串的首地址，p[2]存储第 3 个字符串的首地址*/
   int i;
   for(i=0;i<3;i++)
      printf("%s\n",p[i]); /*输出 3 个字符串 beijing hanghzou nanjing*/
}
```

12.5.2　指针数组的使用

【例题 15】写出下列程序的结果。

```
#include  "stdio.h"
void main()
{  char *st[]={"sun","moon","star","xyz"};
   printf("%s",*(st+1)+1);/*输出结果为 oon*/
}
```

【例题 16】找出 3 个字符串中最大的字符串。

```
#include "string.h"
#include "stdio.h"
void main()
{  char *nation[3]= {"China","Korea","America"};
   char max[80];/*定义 max 数组用来存储最大的字符串*/
   if(strcmp(nation[0],nation[1])>0)/*字符串的比较采用 strcmp 函数*/
```

```
        strcpy(nation,nation[0]);/*字符串的赋值采用 strcpy 函数，并把最大的字符串赋值给 max*/
      else
        strcpy(max,nation[1]);
      if(strcmp(max,nation[2])<0)
        strcpy(max,nation[2]);
      printf("最大的字符串是：%s",max);/*输出 max 的值就是最大字符串的值，即 Korea*/
}
```

【例题 17】下列程序段的输出结果。

```
main()
{  char *st[]={"abcd","efgh","ijkl","mnop"};
   char **p=st;
   p++;
   printf("%s",*p+1);
}
```

例题分析

首先定义了一维指针数组，这个指针数组在内存中的地址为 0X5870，指针数组中存放的是 4 个字符串的首地址，也就是说 st[0]存放的是字符串 abcd 的首地址（即字符 a 的地址 0X2357），st[1]存放的是字符串 efgh 的首地址（即字符 e 的地址 0X3652），st[2]存放的是字符串 ijkl 的首地址（即字符 i 的地址 0X4376），st[3]存放的是字符串 mnop 的首地址（即字符 m 的地址 0X5673），**p=st 表示指针数组 st 在内存中的地址（0X5870）赋给了指针变量 p，也就是说指针 p 指向的是数组 st 的首地址，见图 12-6。执行 p++，也就是指针 p 自加 1 指向了下一个数组元素的位置，即 st[1]的地址，即*p 的值是数组 st[1]的值，而数组 st[1]的值是字符串 efgh 的首地址，所以*p+1 是指字符串 efgh 中 f 的位置，见图 12-7，所以输出的是从 f 开始的字符串即 fgh。

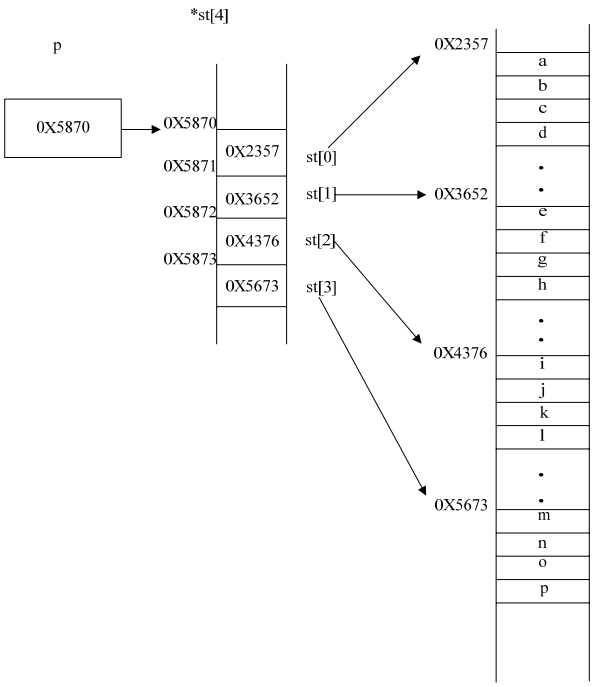


图 12-6　指针数组的表示

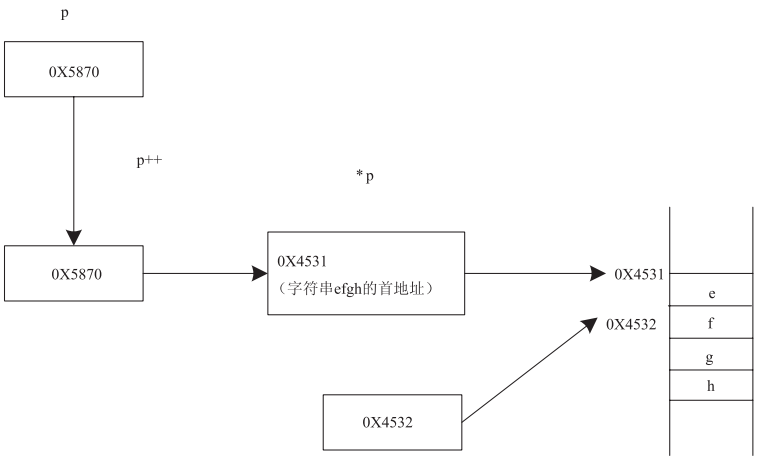


图 12-7　指针 p++运算

【例题 18】有以下程序：（2009 年 3 月）

```
#include <stdio.h>
main()
{  char  *a[ ]={"abcd","ef","gh","ijk"};  int i;
   for（i=0;i<4;i++） printf("%c",*a[i]);
}
```

程序运行后输出的结果是（　　）。

A）aegi　　B）dfhk　　C）abcd　　D）abcdefghijk

例题分析

此题是一个指针数组，指向每个字符串的第一个字符，而且最后输入时用%c 形式输出，所以只能输出第一个字符。所以选 A。

12.5.3　指针的指针的定义

如果一个指针变量存放的又是另一个指针变量的地址，则称这个指针变量为指向指针的指针变量。

格式如下：

类型标识 **指针变量

通过指针访问变量称为间接访问。由于指针变量直接指向变量，所以称为“单级间址”。而如果通过指向指针的指针变量来访问变量则构成“二级间址”，如图 12-8 所示。

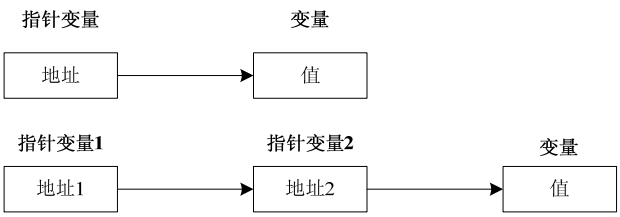


图 12-8　指针的指针

从图 12-9 可以看到，name 是一个指针数组，它的每一个元素是一个指针型数据，其值为地址。name 是一个数据，它的每一个元素都有相应的地址。数组名 name 代表该指针数组的首地址，name+1 是 name[1]的地址，name+1 就是指向

指针型数据的指针（地址）。还可以设置一个指针变量 p，使它指向指针数组元素，p 就是指向指针型数据的指针变量。

怎样定义一个指向指针型数据的指针变量呢？如下：

char **p;

p 前面有两个*号，相当于*(*p)。显然*p 是指针变量的定义形式，如果没有最前面的*，那就定义了一个指向字符数据的指针变量。现在它前面又有一个*号，表示指针变量 p 是指向一个字符指针型变量的。*p 就是 p 所指向的一个指针变量。

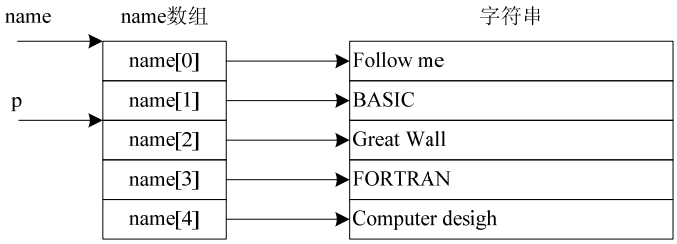


图 12-9　指针数组的内容

如果有：

p=name+2;

printf("%o\n",*p);

printf("%s\n",*p);

则，第 1 个 printf 函数语句输出 name[2]的值（它是一个地址），第 2 个 printf 函数语句以字符串形式（%s）输出字符串“Great Wall”。

【例题 19】使用指向指针的指针。

例题代码

```
main()
{  char *name[]={"Follow me","BASIC","Great Wall","FORTRAN","Computer desighn"};
   char **p;
   int i;
   for(i=0;i<5;i++)
   { p=name+i;
     printf("%s\n",*p);
   }
}
```

【说明】

p 是指向指针的指针变量。

【例题 20】一个指针数组的元素指向数据的简单例子。

例题代码

```
main()
{ static int a[5]={1,3,5,7,9};
  int *num[5]={&a[0],&a[1],&a[2],&a[3],&a[4]};
  int **p,i;
  p=num;
  for(i=0;i<5;i++)
    {printf("%d\t",**p);p++;}
}
```

【说明】

指针数组的元素只能存放地址

【例题 21】指向指针的指针的定义。

```
#include "stdio.h"
void main()
{  char **p;/*定义指向指针的指针变量 p*/
   char *nation[3]={"China","Korea","America"};
   int i;
   for(i=0;i<3;i++)
      printf("%s",*(p+i));
 }
```

12.5.4　指向指针的指针的使用

【例题 22】输出下列程序的结果。

```
#include  "stdio.h"
void main()
{ char **p;
  char *nation[3]= {"China","Korea","America"};
  p=nation+2;/*将 nation[2]的地址赋值给指向指针的指针变量 p*/
  printf("%s",*p+2); /*erica*/
}
```

【考生注意】

*p 表示 nation[2]的值，而 nation[2]的值表示字符串 America 的首地址，即*p=&A，而*p+2 表示字符串 America 中 e 的地址，最后输出*p+2 表示输出以 e 开头的字符串即 erica。

12.5.5　指针变量作为函数参数

一、指针变量为整型或实型

若函数的形参为指针类型，调用该参数时，对应的实参必须是基类型相同的地址值，或者是已指向某个存储单元的指针变量。

【例题 23】编写函数 f(int *a,int *b)，函数中使指针 a 和 b 所指的存储单元中的两个值相减，然后将其值作为函数值返回。在主函数中输入两个数给变量，把变量地址作为实参，传送给对应的形参。

```
#include <stdio.h>
int f(int *a ,int *b)
{ int c;
  sum=*a-*b;
  return c;
}
main()
{ int x,y,z;
  printf("Enter x, y : ");   scanf("%d%d",&x,&y);
```

```
    z=f(&x,&y);
    printf("%d-%d=%d\n",x,y,z);
}
```

在此程序中，主函数调用函数时，系统为 f 函数的形参 a 和 b 开辟两个基类型为 int 类型的临时指针变量，并通过实参&x、&y 的地址传送给它们，这时指针 a 指向变量 x，指针 b 指向变量 y，然后程序的流程转去指向 f 函数。

在 f 函数中，语句“c = *a - * b;”的含义是：分别取指针 a 和 b 所指存储单元中的内容，相加后存入变量 c 中，实际上就是把主函数中 x 变量和 y 变量中的值相加存入变量 c 中。所以，f 函数返回的是主函数中 x 变量和 y 变量中值的差。

由此程序可见，通过传送地址值，可以在被调用函数中对调用函数中的变量进行引用。

【例题 24】编写程序利用指针完成两个数的交换。

```
#include "stdio.h"
void f(int *m,int *n)/*将 p1 的值赋值给 m，将 p2 的值赋值给 n*/
{ int *p;
  p=m;
  m=n;
  n=p;
}
void main()
{ int *p1,*p2;
  scanf("%d%d",p1,p2);
  f(p1,p2);/*将 p1 和 p2 作为参数传递给子函数 f，子函数 f 完成交换功能*/
  printf("%d %d",*p1,*p2);
}
```

二、指针变量为字符型

在前面的章节中讲到了用数组名做函数的实参和形参的问题。在学习指针变量之后就更容易理解这个问题了。数组名就是数组的首地址，实参向形参传送数组名实际上就是传送数组的地址，形参得到该地址后也指向同一数组。在这里所说的数组名作为函数的参数主要是和指针有关的，即数组名和指针之间的参数的关系。

1. 数组元素地址作为实参

当用数组元素地址作为实参时，因为是地址值，所以对应的形参也应当是类型相同的指针变量。

【例题 25】使数组中的每一个元素的值都乘以 2。

```
#include <stdio.h>
void f(int *p)
{ int i;
  *p=*p+1;
}
main()
{ int a[10],k;
  for(i=0;i<10;i++)
   scanf("%d",&a[i]);
  for(i=0;i<10;i++)
   f(&a[i]);
  for(i=0;i<10;i++)
```

```
    printf("%d",a[i]);
}
```

2. 数组名做实参

数组名也可以作为实参传送，数组名本身是一个地址值，因此，对应的形参就应当是一个指针变量，此指针变量的基类型必须与数组的类型一致。在函数中，可以通过此指针变量来引用调用函数中对应的数组元素，从而达到对调用函数中对应的数组输入若干大于或等于 0 的整数，用负数作为输入结束标志；调用另一个函数输出该数组中的数据。

【例题 26】将数组中的每一个元素的值都乘以 2。

```
#include <stdio.h>
void f(int *a,int n)
{  int i;
   for(i=0;i<n;i++)
     *(a+i)=*(a+i)+1;
}
main()
{ int a[10],k;
  for(i=0;i<10;i++)
    scanf("%d",&a[i]);
  f(a,10);
  for(i=0;i<10;i++)
    printf("%d",a[i]);
}
```

当数组名作为实参时，对应的形参除了是指针外，还可以用另外两种形式。对于上例中的调用函数f(a)对应的函数首部可以写成以下 3 种形式。

（1）arrin (int * a)。

（2）arrin(int a[　])。

（3）arrin(int a[M])。

在第（2）和第（3）种形式中，虽然说明的形式与数组的说明相同，但 C 编译程序都将把 a 处理成第一种指针形式。

由此例可见，当传送数组名时，在被调用函数中也同样可用数组元素的形式来引用调用函数中对应的数组元素。应当注意：这只是在形式上的相似，不要误以为整个数组可以传送给被调用函数。在被调用函数中，并没有为与数组名对应的形参另外开辟一串连续的存储单元，而只是开辟了一个指针变量的存储单元。在被调用函数中引用的数组元素就是实参数组中的元素。调用函数只是把数组的首地址传送给形参指针，仍遵循按“值”传送的机制。

关于数组名和指针作为函数之间的实参和形参的关系有以下 4 种关系。

（1）实参是数组名，形参是数组名。

（2）实参是数组名，形参是指针。

（3）实参是指针，形参是数组名。

（4）实参是指针，形参是指针。

【例题 27】写出下列程序的结果。

```
#include "stido.h"
void f(int *x,int *y)
{  ++x;
   --*y;
```

```
}
void main()
{  int x=0;y=3;
   f(&x,&y);
   printf("%d,%d",x,y);
}
```

例题分析

程序在 main()函数中定义了两个变量 x 和 y，然后执行 p(&x,&y)（把变量 x 的地址和变量 y 的地址作为参数传递给子函数 f），在子函数 f 中有两个指针变量来接受从主函数传过来的参数，一个是 x，另一个是 y，也就是说 x=&x，y=&y，然后在子函数中执行++x，也就是把指针变量 x 的值加 1，也就是把&x 的值加 1（因为指针 x 的值是&x），而--*y 就是把指针变量 y 对应的变量的值减去 1（因为指针变量 y 的值就是主函数中的变量 y 的地址，*y 的值为 3，执行完--*y 的值后，结果为 2），子函数执行完后，程序继续执行主函数中的输出语句“printf("%d,%d",x,y);”，由于实参变量 x 的值在此过程中没有发生任何改变，虽然在子函数中执行了++x，并没有对实参变量 x 做任何处理，而变量 y 的值发生了改变，变成了 2。

【例题 28】写出下列程序的结果。

```
#include "stdio.h"
int x;
int p(int *y)
{  ++*y;
   return x-1;
}
void main()
{  int y;
   y=p(&x);
   printf("%d,%d",x,y);
}
```

例题分析

本程序看起来和上面的例题好像没有什么区别，但在这里多了一个全局变量 x，当有一个全局变量在没有对其初始化的时候，它的初值自动赋值为 0，在主函数中把全局变量 x 的地址作为参数传递给子函数中的指针变量 y，然后执行++*y，也就是*y=*y+1，也就是把指针 y 对应的变量 x 的值加 1，所以 x 的值为 1，由于子函数的返回值为 x-1，即子函数的值为 0，主函数中 y 的值为 0。

【例题 29】以下函数按每行 8 个输出数组中的数据。填空。（2009 年 3 月）

```
void fun(int *w, int n)
{  int i;
   for(i=0;i<n;i++)
   {  _______
      printf("%d",w[i]);
   }
   printf("\n");
}
```

下列画线处应填入的语句是（　　）。

A）if(i/8==0)printf("\n");　　B）if(i/8==0)continue;

C）if(i%8==0)printf("\n");　　D）if(i%8==0)continue;

例题分析

此题将指针、数组名、子函数三者结合在一起，但本题主要考查以每行 8 个数输出，答案 C 正确。

12.6　习题

12.6.1　选择题

1. 下列程序的输出结果是（　　）。

```
#include  <stdio.h>
main()
{ int i;
   for(i=1;i<=10;i++)
   { if((i*i>=20)&&(i*i<=100))
       break;
   }
   printf("%d\n",i*i);
}
```

A）49　　B）36　　C）25　　D）64

2. 下列程序的输出结果是（　　）。

```
#include <stdio.h>
char point(char*p){p+=3;}
main()
{ char b[4]={'a','b','c','d'},*p=b;
  point(p);printf("%c\n",*p);
}
```

A）a　　B）b　　C）c　　D）d

3. 若有如下说明和定义语句。

```
char fun(char *);
main()
{ char   s="one",a[5]={0},(*f1)()=fun,ch;
  …
}
```

下列选项中对函数 fun 调用正确的语句是（　　）。

A）(*f1)(a);　　B）f1(s);　　C）fun(&a);　　D）ch=*f1(s);

4. 若有说明 int (*p)[3]，则以下叙述正确的是（　　）。

A）这是一个指针数组

B）p 是一个指针，它只能指向一个包含 3 个 int 类型元素的二维数组

C）p 是一个指针，它可以指向一个一维数组中的任意元素

D）(*p)[3]与*p[3]等价

5. C 语言的定义语句“char *line[5];”含义是指（　　）。

A）line 是一个数组，其数组的每一个元素是指向字符的指针

B）line 是一个指针，指向一个数组，数组的元素为字符型

C）A 和 B 均不对，但它是 C 语言正确的语句

D）C 语言不允许这样定义语句

6. 有以下程序：

```
#include <string.h>
#include <stdio.h>
void f(char *s, char *t)
{ char k;
  k=*s; *s=*t;*t=k;
  s++; t--;
  if(*s) f(s,t);
}
main()
{ char str[10]="abcdefg",*p;
  p=str+strlen(str)/2+1;
  f(p,p-2);
  printf("%s\n",str);
}
```

程序运行后的输出结果是（　　）。

A）abcdefg　　B）gfedcba　　C）gbcdefa　　D）abedcfg

7. 进行如下定义：

```
int (* ptr)();
```

则以下叙述中正确的是（　　）。

A）ptr 是指向一维数组的指针变量

B）ptr 是指向 int 型数据的指针变量

C）ptr 是指向函数的指针，该函数返回一个 int 型数据

D）ptr 是一个函数名，该函数的返回值是指 int 型数据的指针

8. 有以下程序：

```
#include <stdio.h>
void swap(char *x, char *y)
{  char t;
   t=*x;  x=y;  *y=t;
}
main()
{ char s1[]="abc",s2[]="123";
  swap(s1,s2);printf("%s,%s\n",s1,s2);
}
```

程序执行后的输出结果是（　　）。

A）123,abc　　B）abc,123　　C）1bc,a23　　D）321,cba

9. 有以下程序：

```
#include <stdio.h>
main()
{  int a=1,b=3,*p;
   int *p1=&a,*p2=&b;
   *p=(*p1)*(*p2);
   printf("%d\n",*p);
}
```

执行后的输出结果是（　　）。

A）1　　B）2　　C）3　　D）4

10. 有以下程序：

```
#include  <stdio.h>
main()
{  int n,*p=NULL;
   *p=&n;
   printf("Input n:");scanf("%d",&p);printf("output n:");printf("%d\n",p);
}
```

该程序试图通过指针 p 为变量 n 读入数据并输出，但程序有多处错误，下列语句中正确的是（　　）。

A）int n, *p=NULL;　　B）*p=&n;　　C）scanf("%d",&p);　　D）printf("%d\n",p);

11. 下面程序的输出结果是（　　）。

```
#include  <stdio.h>
main()
{  int a[]={1,2,3,4,5,6,7,8,7,10},*p;
   p=a;
   printf("%d\n", *p+8);
}
```

A）0　　B）1　　C）10　　D）9

12. 下列程序的运行结果是（　　）。

```
#include  <stdio.h>
void sub(int *s,int *y)
{ static int m=4;
  *y=s[0];
  m++;
}
void main()
{ int a[]={1,2,3,4,5},k;
  int x;
  printf("\n");
  for(k=0;k<=4;k++)
  { sub(a,&x);
```

```
        printf("%d,",x);
      }
  }
```

A）1,1,1,1,1,　　B）1,2,3,4,5,　　C）0,0,0,0,0,　　D）4,4,4,4,4,

13. 若有定义“char(*p)[6];”，则标识符 p（　　）。

A）是一个指向字符型变量的指针

B）是一个指针数组名

C）是一个指针变量，它指向一个含有 6 个字符型元素的一维数组

D）定义不合法

14. 设 q1 和 q2 是指向一个 float 型一维数组的指针变量，k 为 float 型变量，则下列选项中不能正确执行的语句是（　　）。

A）k=*q1+*q2;　　B）q1=k;　　C）q1=q2;　　D）k=*q1*(*q2);

15. 以下程序的输出结果是（　　）。

```
#include <tdio.h>
void fun(int *a,int i,int j)
{ int  t;
  if(i<j)
  { t=a[i];a[i]=a[j];a[j]=t;
    i++;j--;
    fun(a,i,j);
  }
}
main()
{ int x[]={2,6,1,8},i;
  fun(x,0,3);
  for(i=0;i<4;i++)  printf("%2d",x[i]);
  printf("\n");
}
```

A）1,2,6,8　　B）8,6,2,1　　C）8,1,6,2　　D）8,6,1,2

16. 下列程序的输出结果是（　　）。

```
#include <stdio.h>
void point(char*pt);
main()
{ char b[4]={'m','n','o','p'},*pt=b;
  point(pt);
  printf("%c\n", *pt);
}
void point(char *p)
{p+=3;}
```

A）p　　B）o　　C）n　　D）m

17. 下述程序的运行结果是（　　）。

```
#include <stdio.h>
#include <string.h>
main()
{ char *s1="abDuj";
  char *s2="ABdUG";
  int t;
  t=strcmp(s1,s2);
  printf("%d",t);
}
```

A）正数　　B）负数　　C）零　　D）不确定的值

18. 下列程序的输出结果为（　　）。

```
#include  <stdio.h>
void fun(char *c,int d)
{ *c=*c+1;
  d+=1;
  printf("%c,%c",*c,d);
}
main()
{ char a='F',b='f';
  fun(&b,a);
  printf("%c,%c\n",a,b);
}
```

A）g,GF,g　　B）g,FF,g　　C）G,fF,G　　D）f,gf,g

19. 若有定义“int b[8], *p=b;”，则 p+6 表示（　　）。

A）数组元素 b[6]的值　　B）数组元素 b[6]的地址

C）数组元素 b[7]的地址　　D）数组元素 b[0]的值加上 6

20. 若有说明“int m[3][4]={3,9,7,8,5},(*q)[4];”和赋值语句“q=m;”，则对数组元素 m[i][j]（其中 0<=i<3 , 0<=j<4）值引用正确的是（　　）。

A）(q+i)[j]　　B）*q[i][j]　　C）*(*q[i]+j)　　D）*(*(q+i)+j)

21. 下列程序中的输出结果是（　　）。

```
#include  <stdio.h>
void main()
{ int b[6]={2,4,6,8,10,12};
  int *p=b, **q=&p;
  printf("%d", *(p++));
  printf("%d",**q);
}
```

A）4,4　　B）2,2　　C）4,5　　D）2,4

22. 已有定义 int(*q)()，指针 q 可以（　　）。

A）指向函数的入口地址　　B）代表函数的返回值

C）表示函数的类型　　D）表示函数返回值的类型

12.6.2 填空题

1. 若有定义“float b[15], *p=b;”，数组 b 的首地址为 200H，则 p+13 所指向的数组元素的地址为_____。

2. 执行下列语句段后，x 的值是_____。

```
int *p,x;
x=100;
p=&x;
x=*p+50;
```

3.语句“int(*ptr)();”的含义是_____。

4. 下面 sstrcat()函数的功能是实现字符串的连接，即将 t 所指字符串复制到 s 所指字符串的尾部。例如，s 所指字符串为 abcd，t 所指字符串为 efgh，调用 sstrcat()函数后 s 所指字符串为 abcdefgh。请填空。

```
#include <string.h>
void sstrcat(char*s,char *t)
{ int n;
  n=strlen(s);
  while(*(s+n=__________)){s++; t++; }
}
```

5. 下面程序的功能是：利用指针指向 3 个整型变量，并通过指针运算找出 3 个数中的最大值，输出到屏幕上。请填空。

```
#include <stdio.h>
main()
{ int x,y,z,max,*px,*py,*pz,*pmax;
  scanf("%d%d%d",&x,&y,&z);
  px=&x; py=&y; pz=&z; pmax=&max;
  __________;
  if(*pmax<*py)  *pmax=*py;
  if(*pmax<*pz)  *pmax=*pz;
  printf("max=%d\n",*pmax);
}
```

6. 下面程序段的运行结果是_____。

```
char str[]="ABCD", *p=str;
printf("%d\n", *(p+3));
```

7.下列程序的运行结果是_____。

```
#include  <stdio.h>
#define SIZE 12
void sub(char *, int,int)
main()
{ char s[SIZE];int i;
  for(i=0;i<SIZE;i++)s[i]='A'+i+32;
  sub(s,5,SIZE-1)
  for(i=0;i<SEZE;i++)printf("%c",s[i]);
```

```
  printf("\n");
}
void sub(char *a,int t1, int t2)
{ char ch;
  while(t1<t2)
  { ch=*(a+t1);
    *(a+t1)=*(a+t2);
    *(a+t2)=ch;
    t1++;t2--;}
}
```

8. 若 a=7，b=8，则写出下列程序的执行结果为_____。

```
#include <stdio.h>
void swap(int *p1,int *p2)
{ int p;
  p=*p1;
  *p1=*p2;
  *p2=p;
}
main()
{ int a,b;
  int *p1,*p2;
  scanf("%d%d",&a,&b);
  p1=&a;p2=&b;
  if(a<b)swap(p1,p2);
     printf("\na=%d,b=%d\n",a,b);
}
```

12.6.3　程序设计题

1. 编写函数，求字符串的长度（不能使用 strlen 函数）。
2. 编写函数，将一个长度为 n 的字符串从第 m 个字符开始的全部字符复制成另一个字符串。
3. 用指向指针的指针完成对 5 个字符串进行排序并输出。

第13章 编译预处理

本章主要介绍无参宏定义、带参宏定义、条件编译，根据计算机等级考试二级 C 语言要求，具体如表 13-1 所示。

表 13-1　考试要求

考试知识点	重要性
无参宏定义	★★★
带参宏定义	★★★
条件编译	★★

宏定义是 C 语言提供的 3 种预处理功能其中的一种，这 3 种预处理包括：宏定义、文件包含、条件编译。

13.1　编译预处理

格式如下：

#define 标识符 字符串

其中的标识符就是所谓的符号常量，也称为“宏名”。

例如：# define PI 3.1415926

把程序中出现的 PI 全部换成 3.1415926。

说明：

（1）宏名一般用大写。

（2）使用宏可提高程序的通用性和易读性，减少不一致性，减少输入错误和便于修改。

（3）预处理是在编译之前的处理，而编译工作的任务之一就是语法检查，预处理不做语法检查。

（4）宏定义末尾不加分号。

（5）宏定义写在函数的花括号外边，作用域为其后的程序，通常位于文件的最开头。

（6）可以用#undef 命令终止宏定义的作用域。

（7）宏定义可以嵌套。

（8）字符串中永远不包含宏。

（9）宏定义不分配内存，变量定义分配内存。

【例题 1】写出下列程序的结果。

```
#include  "stdio.h"
#define m=5;
void main()
```

```
{  int y;
   y=m*4;
   printf("y=%d",y);
}
```

例题分析

本题主要考查无参宏定义的使用，由上面的定义得知，y 的值是 20。

13.2　动态存储分配

C 语言允许宏带有参数，在宏定义中的参数为形式参数，而这种分配方式叫做动态存储分配。

13.2.1　动态存储分配

格式：

#define 宏名（参数表） 字符串

说明 ：

（1）宏名和参数的括号间不能有空格。

（2）宏替换只做替换，不做计算，不做表达式求解。

（3）函数调用在编译后程序运行时进行，并且分配内存。宏替换在编译前进行，不分配内存。

（4）函数只有一个返回值，利用宏则可以设法得到多个值。

（5）宏展开使源程序变长，函数调用不会。

（6）宏展开不占运行时间，只占编译时间，函数调用占运行时间（分配内存、保留现场、值传递、返回值）。

【例题 2】写出下列程序的结果。

```
#include  "stdio.h"
#define f(x)  x+x
void main()
{ int y;
  y=f(4)*5;
  printf("y=%d",y);
}
```

例题分析

本题的结果很容易让人觉得答案是 40，其实答案是 24，y 的值是 4+4*5=24，而不是(4+4)*5=40，因为 f(x)表示的是 x+x，而不是(x+x)。

【例题 3】

```
#include "stdio.h"
#define SQ(y) (y)*(y)
main(){
  int a=3,sq;
  printf("input a number: ");
  scanf("%d",&a);
  sq=SQ(a+1)*5;
  printf("sq=%d\n",sq);
}
```

例题分析

答案与上面的例题基本一致，宏定义中定义了 SQ(y)=(y)*(y)，所以是先计算 sq 里面的值再乘以 5 得到最后的结果 sq=80。

【考生注意】

在做题的时候，一定要看清楚宏定义的内容，不要盲目地认为宏定义就是变量或者表达式的基本代替。在考试的时候一定要能够看清楚。

【例题 4】有以下程序：（2009 年 9 月）

```
#include <stdio.h>
#define f(x) x*x*x
main()
{ int a=3,s,t;
  s=f(a+1);t=f((a+1));
  printf("%d,%d\n",s,t);
}
```

程序运行后的输出结果是（　　）。

A）10,64　　B）10,10　　C）64,10　　D）64,64

例题分析

本题主要是考查有参宏定义，并注意括号的意思，f(a+1)=a+1*a+1*a+1 得到 10，而 f((a+1))=(a+1)* (a+1)*(a+1)= 4*4*4= 64。选 A。

13.2.2　条件编译

预处理程序提供了条件编译的功能。可以按不同的条件去编译不同的程序部分，因而产生不同的目标代码文件。这对于程序的移植和调试是很有用的。

条件编译有 3 种形式，下面分别介绍：

1. 第 1 种形式：

```
#ifdef  标识符
    程序段 1
#else
    程序段 2
#endif
```

它的功能是，如果标识符已被 #define 命令定义过则对程序段 1 进行编译，否则对程序段 2 进行编译。如果没有程序段 2（它为空），本格式中的#else 可以没有，即可以写为：

```
#ifdef  标识符
程序段
#endif
```

【例题 5】

```
#define NUM ok
main(){
  struct stu
  {
```

```
    int num;
    char *name;
    char sex;
    float score;
  } *ps;
  ps=(struct stu*)malloc(sizeof(struct stu));
  ps->num=102;
  ps->name="Zhang ping";
  ps->sex='M';
  ps->score=62.5;
  #ifdef NUM
     printf("Number=%d\nScore=%f\n",ps->num,ps->score);
  #else
     printf("Name=%s\nSex=%c\n",ps->name,ps->sex);
  #endif
     free(ps);
}
```

说明：

由于在程序的第 16 行插入了条件编译预处理命令，因此要根据 NUM 是否被定义过来决定编译哪一个 printf 语句。而在程序的第一行已对 NUM 做过宏定义，因此应对第一个 printf 语句做编译，故运行结果是输出了学号和成绩。

在程序的第一行宏定义中，定义 NUM 表示字符串“OK”，其实也可以为任何字符串，甚至不给出任何字符串，写为：

```
#define NUM
```

也具有同样的意义。只有取消程序的第一行才会去编译第二个 printf 语句。读者可上机试做。

2. 第 2 种形式：

```
#ifndef 标识符
  程序段 1
#else
  程序段 2
#endif
```

与第一种形式的区别是将“ifdef”改为“ifndef”。它的功能是，如果标识符未被#define 命令定义过则对程序段 1 进行编译，否则对程序段 2 进行编译。这与第一种形式的功能正相反。

3. 第 3 种形式：

```
 #if 常量表达式
         程序段 1
 #else
         程序段 2
 #endif
```

它的功能是，如常量表达式的值为真（非 0），则对程序段 1 进行编译，否则对程序段 2 进行编译，因此可以使程序在不同条件下，完成不同的功能。

【例题 6】

```
#define R 1
```

```
main(){
  float c,r,s;
  printf ("input a number:  ");
  scanf("%f",&c);
  #if R
    r=3.14159*c*c;
    printf("area of round is: %f\n",r);
  #else
    s=c*c;
    printf("area of square is: %f\n",s);
  #endif
}
```

说明：

例题中采用了第 3 种形式的条件编译。在程序第一行宏定义中，定义 R 为 1，因此在条件编译时，常量表达式的值为真，故计算并输出圆面积。

上面介绍的条件编译当然也可以用条件语句来实现，但是用条件语句将会对整个源程序进行编译，生成的目标代码程序很长，而采用条件编译，则根据条件只编译其中的程序段 1 或程序段 2，生成的目标程序较短。如果条件选择的程序段很长，采用条件编译的方法是十分必要的。

13.3 习题

13.3.1 选择题

1. 下列有关宏替换的叙述中不正确的是（　　）。

A）使用宏定义可以嵌套　　B）宏定义仅仅是符号替换

C）双引号中出现的宏名不替换　　D）宏名必须用大写字母表示

2. #define 能做简单的替代，用宏来替代计算多项式 5*x*x+5*x+5 的值的函数 f，正确的宏定义语句为（　　）。

A）#define f(x) 5*x*x+5*x+5　　B）#define f 5*x*x+5*x+5

C）#define f(a) (5*a*a+5*a+5)　　D）#define (5*x*x+5*x+5) f(x)

3. 在“文件包含”预处理语句的形式中，当#include 后面的文件名用<>（间括号）括起时，寻找被包含文件的方式是（　　）。

A）仅仅搜索当前目录

B）仅仅搜索源程序所在目录

C）在标准目录下查找指定文件

D）先在源程序所在目录搜索，如没找到，再按系统指定的标准目录查找

4. C 语言的编译系统对宏命令是（　　）。

A）在程序运行时进行处理的　　B）在程序连接时进行处理的

C）在源程序中其他 C 语言同时进行编译的　　D）在对源程序中其他成分正式编译之前进行处理的

5. 以下叙述正确的是（　　）。

A）在程序的一行中可以出现多个有效的预处理命令行　　B）使用带参宏时，参数的类型应与宏定义时的一致

C）宏替换不占运行时间，只占编译时间　　D）宏定义不能出现在函数内部

6. 下列有关宏的叙述中不正确的是（　　）。

A）宏名无类型　　B）宏定义不做语法检查

C）宏名必须用大写字母表示　　D）双引号中出现的宏名不进行替换

7. 下述程序的输出结果是（　　）。

```
#include <stdio.h>
#define f(x) x*x;
main()
{ int a=4,b=2,c;
  c=f(a)/f(b);
  printf("%d",c);
}
```

A）2　　B）4　　C）16　　D）32

8. 下述程序的运行结果是（　　）。

```
#include <stdio.h>
#define ADD(x) x+x
main()
{ int m=1,n=2,k=3;
  int s=ADD(m+n)*k;
  printf("sum=%d",s);
}
```

A）sum=18　　B）sum=10　　C）sum=9　　D）sum=25

9. 若有以下宏定义：

```
#define N 2
#define Y(n) ((N+1)*n)
```

则执行语句“z=2*(N+Y(5));”后的结果是（　　）。

A）语句有错误　　B）z=34　　C）z=70　　D）z 无定值

13.3.2　填空题

1. 设有以下宏定义：

```
#define A 2
#define B A+3
```

则执行赋值语句“t=B*2;”后，int 型变量 t 的值为_____。

2. 下述程序的运行结果是_____。

```
#include <stdio.h>
#define A 4;
#define B(x) A*x/2;
main()
{
   float c,a=8.0;
   c=B(a);
```

```
    printf("%f\n",c);
}
```

3. 下述程序的输出结果是_____。

```
#include <stdio.h>
#define p(a) printf ("%d",(int)(a))
#define PRINT(a);printf ("the end")
main()
{ int i,a=0;
  for(i=1;i<5;i++)
    PRINT(a+i);
  printf("\n");
}
```

4. 在宏定义#define Pi 3.14159 中，用宏名 Pi 代替一个_____。
5. 下述程序的运行结果是_____。

```
#define p(a)printf ("%d",a)
main()
{ int j,a[]={1,2,3,4,5,6,7},i=5;
  for(j=3;j>1;j--)
  {switch(j)
    { case 1:
      case 2:p(a[i++]);break;
      case 3:p(a[i--]);
    }
  }
}
```

第14章 结构体和共用体

本章主要介绍结构体类型、结构体变量定义及使用、结构体数组定义及引用、指针与结构体，以及链表的应用，根据等级考试的要求，具体如表 14-1 所示。

表 14-1　考试要求

掌握知识点	重要性
结构体类型的定义	★★
结构体变量	★★
结构体数组的定义与引用	★★
指针与结构体类型	★★★★
链表	★★★★

14.1　结构体类型定义

在 C 语言中，必须先有数据类型，才可以定义属于该类型的变量。结构体类型定义的一般格式如下。

```
struct    结构类型名
  {  数据类型      数据项 1;
     数据类型      数据项 2;
       ……          ……
     数据类型      数据项 n;
};
```

说明：

（1）struct 是关键字，说明是结构体类型，引出结构类型的定义。

（2）“结构类型名”与标识符的命名规则是一样的，最好能见名知意。

（3）所有成员都包含在一对花括号{}内，而每个成员都必须说明是什么类型、什么名称，这与普通变量定义相似。

（4）最后的分号不能省略。

【例题 1】 定义一个日期类型，包括年、月、日 3 个成员。

例题代码

```
    struct  date              /* 日期结构类型：由年、月、日 3 项组成 */
        {   int year;
```

```
        int month;
        int day; };
```

说明：

（1）本书将一个数据项称为结构类型的一个成员或一个分量。

（2）数据类型相同的数据项，既可以逐个逐行定义，也可以合并成一行定义，如上例中日期结构类型，也可以改为如下形式。

```
struct  date
   { int year,month,day;
   };
```

【例题 2】 定义一个反映学生基本情况的结构类型，用以存储学生的相关信息，包括学号、姓名、性别、出生日期、3 门功课的成绩。

例题分析

（1）学号、姓名、性别由一串字符组成，可以用一维字符数组来表示。

（2）出生日期应包括年、月、日，三者不可分，应独立成一个结构类型，且有年、月、日 3 个成员组成。

（3）3 门功课的成绩可以用实型数据结构来表达。

例题代码

```
struct  date              /* 日期结构类型：由年、月、日 3 项组成 */
   { int year,month,day;
   };
/* 学生信息结构类型：由学号、姓名、性别、出生日期、3 门成绩共 7 项组成 */
struct  std_info
  { char no[7];
    char name[9];
    char sex[3];
    struct date birth;
    float score1,score2,score3;
  };
```

说明：

结构类型中的数据项，既可以是基本数据类型，如字符型、实型，也可以是另一个已经定义的结构类型，如上例中“birth”这个数据项，就是一个已经定义的日期结构类型 date。

【思考】

（1）例子中学号、姓名、性别的字符数组实际能存储多少个字符？

（2）有一个商品信息结构类型，包括编号、名称、单价、生产日期、厂家名称、厂家地址，该如何定义？

14.2 结构体变量

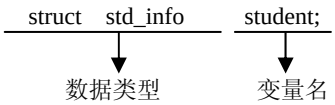
14.2.1 结构体变量的定义

如果用户已经定义好了一个结构类型，如上节【例题 2】的学生信息结构类型 std_info，那么就可以像定义基本整型、实型、字符型变量一样定义结构类型变量了。

定义结构类型变量的方法有两种。

（1）间接定义法——先定义结构类型，再定义结构变量

例如：利用上节【例题 2】中定义的学生信息结构类型 std_info，定义一个相应的结构变量 student 为：



说明：

①使用间接定义法定义结构变量时，必须同时指明结构类型名。

②结构变量 student：拥有结构类型 std_info 的全部 7 个成员，其中 birth 成员是一个日期结构类型，它又由 3 个成员构成。

（2）直接定义法——在定义结构类型的同时，定义结构变量

例如：结构变量 student 的定义可以改为如下形式：

```
struct  std_info
    {……
    } student;
```

直接法定义结构变量的一般格式如下：

```
struct  [结构类型名]
  { ……
  } 结构变量表;
```

如果结构变量表中不止一个变量，用逗号分隔开。

说明：

（1）结构类型与结构变量是两个不同的概念，其区别如同 int 类型与 int 型变量，在高级语言中，相当于是类与实例的关系。

（2）结构类型中的成员，可以与程序中的其他变量同名，但它们代表不同的对象，互不干扰，引用方法也不相同。

14.2.2　结构体变量的引用与初始化

1. 结构体变量的引用

对于结构变量，不能像基本类型变量那样，直接通过变量名来整体访问，而是要通过成员运算符“.”逐个访问其成员，且访问的格式为：

结构变量.成员

例如：上例中的 student.score1，引用结构变量 student 中的 score1 成员；student.sex 引用结构变量 student 中的 sex 成员，等等。

如果某成员本身又是一个结构类型，则只能通过多级的分量运算，对最低一级的成员进行引用。此时的引用格式扩展为：

结构变量.成员.子成员……最低一级子成员

例如：引用结构变量 student 中的 birth 成员的格式分别为：

student.birth.year

student.birth.month

student.birth.day

说明：

（1）对最低一级成员，可像同类型的普通变量一样，进行相应的各种运算。

（2）既可引用结构变量成员的地址，也可引用结构变量的地址。

例如，&student.name，&student。

2. 结构体变量的初始化

结构变量初始化的格式，与一维数组相似，一般格式如下：

结构变量={初值表}

不同的是：如果某成员本身又是结构类型，则该成员的初值为一个初值表。

例如，【例题 2】中的日期初始化：

student={"060401","小明","男",{1986,12,20},78,85,82};

【考生注意】初始化时初值的数据类型、顺序，应与结构变量中相应成员的数据类型及顺序位置相一致，否则会出错。

【例题 3】有以下程序：（2009 年 9 月）

```
#include <stdio.h>
typedef struct
{ int num;double s}REC;
void fun1( REC x )
{  x.num=23;x.s=88.5;
}
main()
{ REC a={16,90.0};
  fun1(a);
  printf("%d\n",a.num);
}
```

程序运行后的输出结果是_____。

例题分析

本题主要是把函数和结构体结合起来考查考生，由于子函数的返回值是 void，所以无论子函数进行什么运算，都不会影响到主函数，所以，此题的输出结果是 16。

14.2.3 结构体数组的定义与引用

所谓结构体数组，就是说数组的每个元素都是结构体类型，且每个元素都含有相同的成员。这就是高级语言中所说的“表”了。

【例题 4】利用【例题 2】中定义的结构类型 std_info，定义一个结构数组 a，用于存储和显示 3 个学生的基本情况（采用从键盘输入的方式存储到数组中）。

例题代码

```
#include "struct.h"
struct  std_info  a[3];    /* 定义一个外部结构数组 a[3] */
main()
  { int i;
    for(i=0;i<3;i++)      /* 从键盘输入 3 个学生的基本信息 */
       { printf("No:");      scanf("%s",a[i].no);
         printf("Name:");     scanf("%s", a[i].name);
```

```
    printf("Sex:");      scanf("%s", a[i].sex);
    printf("Birthday:"); scanf("%d%d%d",&a[i].birth.year,
           &a[i].birth.month, &a[i].birth.day);
    printf("Score1,Score2,Score3:");
    scanf("%f%f%f",&a[i].score1,&a[i].score2,&a[i].score3);
    }
    /* 打印表头: "  "表示 1 个空格字符 */
  printf("No.   Name    Sex Birthday      Score1 Score2 Score3\n");
  for(i=0; i<3; i++)          /* 输出 3 个学生的基本情况 */、
     { printf("%-7s",a[i].no);
       printf("%-9s",a[i].name);
       printf("%-4s",a[i].sex);
       printf("%d-%d-%d",a[i].birth.year,a[i].birth.month, a[i].birth.day);
       printf("%-6.1f%-6.1f%-6.1f\n",a[i].score1,a[i].score2,a[i].score3);
     }
}
```

14.2.4　指向结构体变量的指针

指向结构体变量的指针该如何定义及使用呢？下面我们通过例子来学习它。

【例题 5】使用结构指针来访问结构体变量的各个成员。

例题代码

```
#include "struct.h"
/* 定义并初始化一个外部结构变量 student */
struct  std_info  student={"060401","小明","男",{1986,12,20}, 78, 85, 82};
main()
{ struct std_info *p=&student; /* 定义指针变量 p，并指向结构变量 student*/
  printf("No: %s\n",p->no);
  printf("Name: %s\n",p->name);
  printf("Sex: %s\n",p->sex);
  printf("Birthday: %d-%d-%d\n", p->birth.year,p->birth.month, p->birth.day);
  printf("Score1,Score2,Score3:%.1f,%.1f,%.1f\n", p->score1,p->score2, p->score3);
}
```

例题分析

（1）定义指向结构类型的指针变量，与定义指向基本类型的指针变量是完全一样的，遵循“同类型”原则。

（2）通过指向结构变量的指针来访问结构变量的成员，与直接使用结构变量的效果是一样的。

（3）假设指针变量 p 已指向结构变量 var，则以下 3 种形式等价。

- var.成员　/* “.”是结构体成员运算 */。
- p->成员　/* “->”是结构体指针的指向运算，由减号和大于号组成，中间不能有空格 */。
- (*p).成员　/* “*p”外面的括号不能省略！*/。

14.2.5 指向结构体数组的指针

结构指针变量不但可以指向一个结构体变量，也可以指向结构体数组中的元素。结构指针可以引用结构体变量的成员，那该如何引用结构体数组元素的成员呢？

下面我们通过例子来进一步分析。

【例题 6】利用指向结构体数组的指针来访问结构体数组。

例题代码

```
#include "struct.h"
/* 定义并初始化一个外部结构数组 a */
struct  std_info  a[3]=
{{"060401","小明","男",{1986,12,20},78,85,82},
{"060402","张玉英","女",{1985,10,10},98,95,92},
{"060403","李良","男",{1986,1,12},88,60,70}};
 main()
  { struct std_info *p=a; /* 定义指针变量 p，并指向结构数组 a */
    int i;
    /* 打印表头: "  "表示 1 个空格字符 */
   printf("No.   Name    Sex Birthday      Score1 Score2 Score3\n");
   for(i=0; i<3; i++,p++)          /* 输出 3 个学生的基本情况 */
   { printf("%-7s",p->no);
     printf("%-9s", p->name);
     printf("%-4s", p->sex);
     printf("%d-%d-%d", p->birth.year,p->birth.month, p->birth.day);
     printf("%-6.1f%-6.1f%-6.1f\n", p->score1,p->score2, p->score3);
   }
}
```

14.2.6 链表

链表是一种常用的、能够实现动态存储分配的数据结构，本书从应用角度考虑，只涉及单链表，如图 14-1 所示。

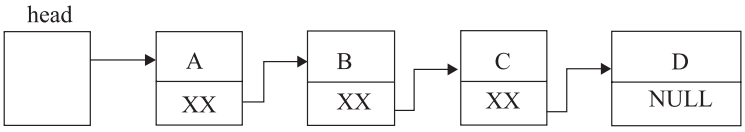


图 14-1　单链表

在链表上，可以非常方便地增加一个结点或删除一个结点。

head：链表的头指针，存放的是地址，指向链表的第一个元素（结点）。

链表结点：分成两部分，存放实际数据的数据域和指向后继结点的指针域。

链表一环扣一环地将多个结点链接在一起，即头指针 head 指向第 1 个结点，第 1 个结点又指向第 2 个结点，……，直到链表的最后一个结点。我们一般称链表的第 1 个结点为“首结点”，链表的最后一个结点为“尾结点”（尾结点的后继指针一般为“NULL”，表示链表不再有后继结点）。当链表中没有任何一个结点时称为空链表，此时链表头指针为“NULL”，

在单链表中，一个结点的后继结点位置存储在它所包含的某个指针成员中，单链表的每个结点在内存中的存放位置是可以任意的。如果某个算法要对链表做某种处理，必须从链表的首结点开始，顺序处理链表的结点。

链表与数组相比较，主要的区别如下。

（1）数组的元素个数在数组定义时指定，元素个数是固定的；链表结点的存储空间是在程序执行时由程序动态向系统申请的，链表的结点个数可按需要增减。

（2）数组的元素顺序关系由元素在数组中的下标确定，链表结点顺序关系由结点的指针实现。

（3）在数组中插入或删除结点，要移动部分结点的存储位置。在链表中插入或删除结点，不要移动结点，仅改变结点的指针指向即可。

动态数据结构中的数据对象是一种结构变量，有存储数据信息的成员，还有存储指针的成员。最简单的情况是结构包含有指向与自己同样结构的指针。例如：

```
struct intNode                /* 整数链表结点类型 */
{  int value ;                /* 存放整数 */
   struct intNode *next ;     /* 存放后继结点的指针 */
 };
```

说明：

在结构类型 intNode 中，有两个成员，value 和指向后继结点的指针成员 next。其中指针成员 next 能指向的结构类型就是正在定义的 intNode。这种结构又称为自引用结构。利用指针成员 next，可把一个 intNode 类型的结构与另一个同类型的结构链接起来，用于描述动态数据结构中两数据对象之间的关系。

动态数据结构中的变量称为动态变量，动态变量能由程序调用库函数，显式地随意生成和释放（消去）。

在 C 语言中，有 malloc 和 calloc 两个库函数用于生成动态变量，另有一个库函数 free 用于释放动态变量。

1. malloc(size)

功能：从系统提供的动态存储区域中分配 size 个字节的连续空间，函数返回该连续空间的开始地址。如果因动态存储区域已分配完，而不能满足这次申请要求，函数将返回 0（NULL）值。

说明：因函数 malloc()的返回值是无类型指针值，程序可将该返回值强制转换成某种特定的指针类型。

2. calloc(n, size)

功能：在动态存储区中分配 n 个 size 个字节的连续空间，函数返回该连续空间的开始地址；如果分配不成功，则返回 0。

例如：申请分配能存储 100 个整数的存储块。

```
p= (int *)malloc(100 * sizeof(int));
q= (int *)calloc(100, sizeof(int));
```

3. free(ptr)

功能：释放由 ptr 所指向的存储块。

说明：ptr 的值必须是调用函数 malloc()或 calloc()申请到的连续存储空间的起始地址。

单链表的基本操作主要包括建立空的单链表、创建一个结点、遍历单链表、查找指定值的结点、往单链表插入一个结点、从单链表删除一个结点等。下面以学生单链表为例，说明各种操作的实现算法。

14.2.7　链表的创建

链表的建立过程是从空链表（没有链表结点）出发，逐渐插入链表新结点的过程。

【例题 7】创建一个空链表。

例题分析

（1）定义结点结构。

（2）要使以变量 head 为头指针的链表为空链表，让 head 取 NULL 值即可。

例题代码

```
struct grade              /* 学生链表结点结构类型 */
{  char  no[7];           /* 学号 */
   char name[9]           /* 姓名 */
   int  score;            /* 成绩 */
   struct  grade  *next;  /* 指针域 */
}*head= NULL;             /* head 是链表的头指针 */
```

【例题 8】编写一个 create()函数，按照规定的结点结构，创建一个单链表（链表中的结点个数不限）。

例题分析

函数首先调用 malloc()申请新结点（New）的空间，然后输入结点数据域的数据项，并将指针域置为空（链尾标志），最后将新结点插入到链表尾（Tail）。对于链表的第一个结点，还要设置头指针变量（head），用做函数返回值。

例题代码

```
#define   NULL  0
#define   LEN   sizeof(struct grade)     /* 定义结点长度 */
struct  grade                            /* 定义结点结构 */
   { char  no[7];                        /* 学号 */
     char name[9]                        /* 姓名 */
     int  score;                         /* 成绩 */
     struct  grade  *next;               /* 指针域 */
      };
/* create()函数：创建一个具有头结点的单链表 */
   struct grade *create( void )
        { struct grade *head=NULL, *new, *tail;
          int count=0;                   /* 链表中的结点个数（初值为 0）*/
          for(  ;   ;  )                /* 缺省 3 个表达式的 for 语句 */
         { new=(struct grade *)malloc(LEN); /* 申请一个新结点的空间 */
/* 1. 输入结点数据域的各数据项 */
          printf("Input the number of student No.%d(6 bytes): ", count+1);
          scanf("%6s", new->no);
          if(strcmp(new->no,"000000")==0)       /* 如果学号为 6 个 0，则退出 */
               { free(new);              /* 释放最后申请的结点空间 */
                 break;                  /* 结束 for 语句 */
               }
          printf("Input the name of the student No.%d: ", count+1);
          scanf("%s", new->name);
          printf("Input the score of the student No.%d: ", count+1);
          scanf("%d", &new->score);
          count++;                      /* 结点个数加 1 */
          new->next=NULL;               /* 2. 置新结点的指针域为空 */
/*3. 将新结点插入到链表尾，并设置新的尾指针*/
```

```
            if(count==1)
            head=new;            /* 是第一个结点，置头指针 */
            else
            tail->next=new;      /* 非首结点，将新结点插入到链表尾 */
            tail=new;            /* 设置新的尾结点 */
               }
               return(head);
}
```

【思考】

（1）“new=(struct grade *)malloc(LEN);”语句申请新结点时，为什么要强制转换？

（2）“scanf("%6s", new->no);”语句加入宽度 6 有什么用处？

14.2.8　链表的插入

往单链表插入一个结点是建立单链表的主要操作之一，它又可按结点插入在单链表中的位置不同分两种情况：可以插在链表的最前面（作为首结点）、可以插在指定的结点之后。

（1）在原来的首结点之前插入一个新结点

插入的结点将成为链表新的首结点。这个工作包括将链表原来的首结点接在新插入的结点之后，并修改链表的头指针，使链表的头指针指向新插入的结点。设指针 p 指向新申请到的结点地址，实现这个功能的代码如下。

```
p->next=head;          /* 新结点指向原首结点地址，使原首结点成为后继结点 */
head=p;                /* 头指针指向新结点，使新结点成为首结点 */
```

（2）在指定的结点之后插入新结点

对于单链表，一般要求将新结点插在指定结点之后。假设指针 w 指向指定的结点（w 不为 NULL），指针 p 指向要插入的结点。实现这个功能的代码如下。

```
p->next=w->next;       /* 新结点指向指定结点的后继结点 */
w->next=p;             /* 新结点成为指定结点的后继结点 */
```

【例题 9】编写一个 insert()函数，完成在单链表的第 i 个结点后插入 1 个新结点的操作。当 i=0 时，表示新结点插入到第一个结点之前，成为链表新的首结点。

例题分析

通过单链表的头指针，首先找到链表的第一个结点；然后顺着结点的指针域找到第 i 个结点，最后将新结点插入到第 i 个结点之后。

例题代码

```
struct grade *insert(struct grade *head, struct grade *new, int i)
   { struct grade *pointer;
if(head==NULL)            /* 将新结点插入到 1 个空链表中 */
head=new, new->next=NULL;
else          /* 非空链表*/
if(i==0)      /* 使新结点成为链表新的首结点 */
  new->next=head,  head=new;
else                      /* 其他位置 */
     {  pointer=head;
```

```
    /* 查找单链表的第 i 个结点（pointer 指向它）*/
        for(;pointer!=NULL&&i>1; pointer=pointer->next, i--);
            if(pointer==NULL)           /* 越界错 */
                printf("Out of the range, can't insert new node!\n");
            else       /* 一般情况：pointer 指向第 i 个结点 */
                new->next=pointer->next, pointer->next=new;
        }
    return(head);
  }
```

【思考】

编写在学生链表的末尾接上一个新学生结点的函数。

14.2.9 链表的删除

要将一结点从链表中删除，首先要在链表中查找该结点，若未找到，则不用做删除工作。在删除时还要考虑两种不同的情况。

（1）如果删除的是首结点，则只要修改链表头指针即可。

（2）如果删除的结点不是链表的首结点，则要更改删除结点的前驱结点的后继指针。

【例题 10】编写在学生链表中删除指定学号的结点的函数。

例题分析

删除特定的结点必须先做查找工作。在查找过程中，必须记住当前结点的前驱结点，待查找结束时，不仅找到了要删除的结点，同时得到要修改前驱结点的后继指针。

例题代码

```
struct grade *delStu (struct grade *head, char num[]) /* num 为要查找的学号 */
{   struct grade *w, *p;
    if (head == NULL)
    return NULL;
    p = head;                                    /* 让 p 指向链表的首结点 */
    while (p != NULL && strcmp(p->no,num))       /* p 不是要找的结点 */
 {  w=p;
    p=p->next;          /* w 指向前驱结点，p 指向下一个结点 */
     }
    if(p!=NULL)             /* 找到 */
     {  if(p==head)
                head=p->next;    /* 首结点即是要找到的结点，则删除首结点，改链表头指针 */
        else
                w->next=p->next; /* 修改前驱结点的后继指针 */
     printf("学号 %s  已被删除\n", num);
     free(p);      /* 释放被删结点的存储空间 */
    }
  else printf("学号 %s  找不到! \n", num);
  return head;    /* 返回头指针 */
}
```

14.3　共用体

共用体类型变量的定义分两步：共用体类型定义与变量定义。

一、共用体类型定义：共用体是用户自定义类型，须先定义

```
一般形式:union  结构体类型名{
                        成员名及类型表
                        }
```

例如：

```
union data{
          int i;
          char c;
          float f; }
 struct date{
          int year;
          int month;
          int day; }
```

二、共用体变量的定义

1. 定义共用体类型后定义共用体变量。

如接上例：union data a1,a2,a3;

2. 定义类型同时定义变量“union data{

int i;char c;float f;}b1,b2;”。

3. 无名共用体变量“union {

int i;char c;float f;}b1,b2;”。

【考生注意】

共用体与结构体的区别。

1. 结构体变量所占的存储空间为各成员所占存储空间之和，而共用体的成员共享存储单元，因此共用体变量占用的存储单元为其最长的成员所占的存储单元。如上例 data 类型的共用体变量 a1，a2 占用的存储单元为 4 个字节，若把 data 定义为结构体变量则须分配 7 个字节空间。

2. 结构体各成员占用连续的存储单元，结构体变量的首地址与第一个成员的首地址相同；共用体变量的首地址与各成员的首地址一样：

&a1=&a1.i=&a1.c=&a1.f

三、共用体变量的引用

除整体赋值外，C 语言规定对共用体变量不能整体引用，只能逐个引用共用体变量的各成员。共用体变量成员的引用方法：共用体变量名.成员变量名，若某个成员还是结构体或共用体成员，则对此成员再逐个引用其成员（同结构体）。

说明：

1. 共用体中的各成员可以像使用同类型的变量一样使用。

2. 可引用共用体变量的地址也可引用各成员的地址：&a1.i。

3. 允许一个有值的共用体变量给另一个整体赋值：b1=a1。

例如：union data a;

a.i=5;

printf("%d\n",a.i);

四、共用体数据的特点

1. 共用体各成员在某一时刻只能有一个成员有意义，不能同时起作用，有意义的成员是最后一次赋值的成员。

例如：“union data a;

a.i=5;a.f=3.14;a.c='a';”其有效的成员是 a.c。

2. 共用体各成员的首地址与共用体变量的首地址相等。

3. 可以对共用体变量进行整体赋值。

4. 对共用体变量进行初始化只能对第一个成员进行初始化。

例如：可以用"union data a={1};"

但不能用"union data a={1,'c',3.14};"。

5. 共用体变量可以作为函数参数，可以使用共用体指针变量。

6. 结构体的某成员可以是共用体成员，而共用体的某成员也可以是结构体成员。

例如：struct test{int a;char c[10];uniondata d;};

union test{int a;char c;struct student s;};

下面我们可以看一个共用体的例子。

【例题 11】判断下列程序的运行结果。

```
main()
{ union  v_val {
  int ival;
  float fval;
  char *pval;} x={100},y,*px;
  px=&x;
  printf("%d***%d\n",x.val,p->val);
  y.ival=x.ival;
  printf("%d###\n",y.val);
  p->pval="china";
  printf("%s***%d\n",(*px).pval,x.ival);
  y.fval=3.14;
  printf("%f+++%d\n",y.fval,y.ival);
}
【结果】 100***100
        100###
        china***424
        3.140000+++-2126
main()
{ union{
  int x;
  char c;} a;
  a.x=0x4100;
  printf("%c",a.y);
}
【结果】  A
```

14.4 习题

14.4.1 选择题

1. 定义下述结构体数组：

```
struct stu
{ int num;
  char name[20];
}x[5]={1,"LI",2,"zhao",3,"wang",4,"zhang",5,"liu"};
for(i=1;i<5;i++)
   printf("%d%c",x[i].num,x[i].name[2]);
```

以上程序段的输出结果为（　　）。

A）2A3N4A5U　　B）1I2A3H4I　　C）1A2N3A4U　　D）2H3A4H5I

2. 定义下述结构体（联合）数组：

```
struct st
{  char name[15];
   int age;
}a[10]={"zhao",14,"wang",15,"liu",16,"zhang",17};
```

执行语句 printf("%d,%c",a[2].age,*(a[3].name+2))的输出结果为（　　）。

A）15,A　　B）16,H　　C）16,A　　D）17,H

3. 若有下述结构体定义：

```
struct stu{int num;
   char sex;
   int age;
}a1,a2;
```

则下述语句中错误的是（　　）。

A）printf("%d,%c,%d,",al);　　B）a2.age=al.age;　　C）al.age++;　　D）al.num=5;

4. 若有以下说明和语句，则对结构体变量 st 中成员 i 的引用方式不正确的是（　　）。

```
struct  stu
{ int i;
  int name;
}st,*p;
p=&st;
```

A）st.i　　B）*p.i　　C）(*p).i　　D）p->i

5. 有如下定义：

```
struct date
{
   int year,month,day;
}
struct worklist
{
```

```
   char name[20];
   char sex;
   struct date birthday;
}person;
```

对结构体变量 person 的出生年份进行赋值时，下面正确的赋值语句是（　　）。

A）year=1958　　B）birthday.year=1958

C）person.birthday.year=1958　　D）person.year=1958

6. 有一个名为 init.txt 的文件，内容如下：

```
#define hdy(a,b) a/b
#define print(y) printf("y=%d/n",y)
```

若有以下程序：

```
#include  "init.txt"
main()
{ int a=1,b=2,c=3,d=4,k;
  k=hdy(a+c,b+d);
  print(k);
}
```

下列针对该程序的叙述中正确的是（　　）。

A）编译错误　　B）运行错误　　C）运行结果为 y=0　　D）运行结果为 y=6

7. 在位运算中，操作数每右移两位，其结果相当于（　　）。

A）操作数乘以 2　　B）操作数除以 2　　C）操作数除以 4　　D）操作数乘以 4

8. 若 x=10010111，则表达式(3+(int)(x))&(-3)的运行结果是（　　）。

A）10011000　　B）10001100　　C）10101000　　D）10110000

9. 已知有如下结构体：

```
struct sk
{  int a;
   float b;
}data *p;
```

若有 p=&data，则对 data 的成员 a 引用正确的是（　　）。

A）(*p).data.a　　B）(*p)a;　　C）p−>data.a　　D）p.data..a

10. 有以下程序：

```
#include <stdio.h>
struct stu
{  int num;
   char name[10];
   int age;};
 void fun(struct stu *p)
 { printf("%s\n",(*p).name);}
 main()
 {  struct stu students[3]={{9801,"Zhang",20},{9802,"Wang",19}{9803,"Zhao",18}}
    fun(students+2);
}
```

输出的结果是（　　）。

A）Zhang　　B）Zhao　　C）Wang　　D）18

14.4.2　填空题

1. 下述程序运行后的输出结果是_____。

```
struct NODE
{int num:  struct  NODE *next;
};
main()
{ struct NODE S[3]={{1,NULL},{2,NULL},{3,NULL}},*p*q,*r;
  int sum=0;
  s[0].next=s+1; s[1].next=s+2; s[2].next=s;
  p=s;q=p->next;r=q->next;
  sum+=q->next->num;sum+=r->next->num;
  printf("%d/n",sum);
}
```

14.4.3　程序设计题

1. 定义一个日期结构变量，计算该日期是本年度的第几日？

2. 某学习小组有 5 名学生，每个人的信息包括学号、姓名和成绩，要求从键盘输入他们的信息，并求出平均成绩，输出最高成绩者的所有信息。

3. 编写一个 edit()函数，用于修改学生成绩单链表中学号为 no 的学生成绩。

第15章 位运算

本章主要介绍位运算符、按位与运算、按位或运算、按位异或运算、求反运算、左移运算及右移运算，结合计算机等级考试二级 C 语言的要求，具体如表 15-1 所示。

表 15-1　考试要求

考试知识点	重要性
位运算符	★★
按位与运算	★★
按位或运算	★★
按位异或运算	★★
求反运算	★★
左移运算	★★
右移运算	★★

15.1　位运算符

位运算是指二进制位的运算，在系统软件中处理二进制的问题需要左移或者右移，而这就是位运算。

C 语言提供了 6 种位运算符。

&	按位与
|	按位或
^	按位异或
~	取反
<<	左移
>>	右移

15.2　位运算符和位运算

1. 按位与运算

按位与运算符&是双目运算符。其功能是参与运算的两数对应的二进位相与。只有对应的两个二进位均为 1 时，结果位才为 1，否则为 0。参与运算的数以补码形式出现。

例如，9&5 可写算式如下：

```
  00001001          （9 的二进制补码）
&00000101           （5 的二进制补码）
  00000001          （1 的二进制补码）
```

可见 9&5=1。

按位与运算通常用来对某些位清零或保留某些位。例如，把 a 的高 8 位清零，保留低 8 位，可做 a&255 运算（255 的二进制数为 0000000011111111）。

【例题 1】

```
#include  "stdio.h"
void main()
{
   int a=9,b=5,c;
   c=a&b;
   printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

2. 按位或运算

按位或运算符|是双目运算符。其功能是参与运算的两数对应的二进位相或。只要对应的两个二进位有一个为 1 时，结果位就为 1。参与运算的两个数均以补码出现。

例如，9|5 可写算式如下：

```
 00001001
|00000101
 00001101          （十进制为 13）可见 9|5=13
```

【例题 2】

```
# include "stdio.h"
void main()
{
   int a=9,b=5,c;
   c=a|b;
   printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

3. 按位异或运算

按位异或运算符^是双目运算符。其功能是参与运算的两数对应的二进位相异或，当两对应的二进位相异时，结果为 1。参与运算的两个数仍以补码形式出现，例如 9^5 可写成算式如下：

```
 00001001
^00000101
 00001100          （十进制为 12）
```

【例题 3】

```
# include "stdio.h"
void main()
{
   int a=9;
   a=a^5;
   printf("a=%d\n",a);
}
```

4. 求反运算

求反运算符～为单目运算符，具有右结合性。其功能是对参与运算的数的各二进位按位求反。

例如：～9 的运算为：

~(0000000000001001)结果为 1111111111110110。

5. 左移运算

左移运算符<<是双目运算符，其功能把<<左边的运算数的各二进位全部左移若干位，由<<右边的数指定移动的位数，高位丢弃，低位补 0。

例如：

a<<4

指把 a 的各二进位向左移动 4 位。如 a=00000011（十进制 3），左移 4 位后为 00110000（十进制 48）。

6. 右移运算

右移运算符>>是双目运算符，其功能是把>>左边的运算数的各二进位全部右移若干位，由>>右边的数指定移动的位数。

例如：

设 a=15，

a>>2

表示把 000001111（十进制 15）右移为 00000011（十进制 3）。

应该说明的是，对于有符号数在右移时，符号位将随同移动。当为正数时，最高位补 0，而为负数时，符号位为 1，最高位是补 0 或是补 1 取决于编译系统的规定。Turbo C 和很多系统规定为补 1。

【例题 4】若有以下程序段：（2009 年 9 月）

```
int r=8;
printf("%d\n",r>>1);
```

输出结果是

A）16　　B）8　　C）4　　D）2

例题分析

本题实际上考查右移 1 位的结果，除了上面所讲的换成二进制的做法外，还可以直接进行运算，右移 n 位，就是除以 2 的 n 次方，而左移 n 位，就是乘以 2 的 n 次方，所以，此题的答案为 C。

15.3 习题

选择题

1. 有以下程序：

```
main()
{  unsigned char a=2,b=4,c=5,d;
   d=a|b;d&=c;printf("%d\n",d);}
```

程序运行后的输出结果是（　　）。

A）3　　B）4　　C）5　　D）6

2. 若变量 s 已正确定义，则以下语法的输出结构是（　　）。

```
s=32;   s^=32;   printf("%d",s);
```

A）-1　　B）0　　C）1　　D）32

3. 设有以下语句：

```
char x=1,y=2,z;
z=x^y<<2;
```

执行后，z 的二进制值为（　　）。

A）00010100　　B）00011011　　C）00011100　　D）00011000

4. 有以下程序：

```
#include <stdio.h>
main()
{ into a=5,b=1,t;
  t=(a<<2)|b;
  printf("%d\n",t)
}
```

程序运行后的输出结果是（　　）。

A）21　　B）11　　C）6　　D）1

5. 有以下程序：

```
#include <stdio.h>
main()
{  char a=4;
   printf("%d\n",a=a<<1);
}
```

程序运行的结果是（　　）。

A）40　　B）16　　C）8　　D）4

6. 在 16 位机上，以下程序的运行结果是（　　）。

```
#inclube <stdio.h>
main()
{
   struct st
  { unsigned a:10;
    unsigned b:12;
    unsigned c:2;
  }x;
  printf("%d\n",sizeof(x));
}
```

A）2　　B）3　　C）24　　D）不能通过编译

7. 在 16 位机上，以下程序的运行结果是（　　）。

```
#inclube  <stdio.h>
main()
{
  struct st
  {
     unsigned a:20;
  }x;
  printf("%d\n",sizeof(x));
}
```

A）1　　B）2　　C）3　　D）不能通过编译

第16章 文件

本章主要介绍文件的概念，文件类型指针及文件操作函数。结合 C 语言教学及计算机等级考试和程序员考试中的要求，具体如表 16-1 所示。

表 16-1　考试要求

掌握知识点	重要性
文件	★★
文件类型指针	★★
文件的打开与关闭	★★★
文件的读写函数	★★★
出错的检测	★

16.1　C 语言文件的概念

16.1.1　文件与文件名

所谓“文件”是指一组相关数据的有序集合。实际上在前面的各章中我们已经多次使用了文件，例如，源程序文件、目标文件、可执行文件、库文件 （头文件）等。这个数据集有一个名称，叫做文件名。文件名的一般结构为：主文件名[.扩展名]。

文件命名规则遵循操作系统的约定，一般来说由英文字母（大小写均可）、数字 0～9、下画线组成，其中主文件名的最大长度不超过 8，扩展名不超过 3。

16.1.2　文件分类

文件的类型有很多种，下面从不同的角度对文件进行分类。

（1）根据文件的内容，可分为程序文件和数据文件，程序文件可以分为源文件、目标文件和可执行文件。如在 C 语言编译环境下的后缀名为 c 的文件、obj 文件和 exe 文件。

（2）根据文件的组织形式，可以分为顺序存取文件和随机存取文件。

（3）根据文件编码的方式，文件可分为 ASCII 码文件和二进制码文件两种。ASCII 文件也称为文本文件，这种文件在磁盘中存放时每个数字对应一个字节，用于存放对应的 ASCII 码。二进制文件把内存中的数据，原样输出到磁盘文件中。

假设有一个整数 5678，如果按照二进制码的形式存储，那么需要 2 个字节就够用了；但是按 ASCII 码形式存储，由于每位数字都要用 1 个字节，所以总共需要 4 个字节空间，如图 16-1 所示。

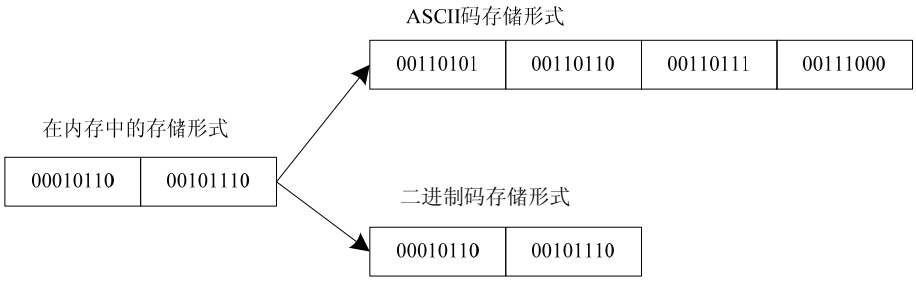


图 16-1　数据在内存中的存储

那么以上这两种文件有什么不同呢？从上图可以观察到用 ASCII 码形式存储，1 个字节存储 1 个字符，因而便于对字符进行逐个处理（二进制码形式与 ASCII 码之间的转换），但一般占用存储空间较多。而采用二进制码存储，可以节省存储空间和转换空间，但 1 个字节并不对应 1 个字符，不能直接输出字符形式。一般中间结果数据需要暂时保存在外存且后面又需要输入到内存中，所以常使用二进制文件保存。

由于 C 语言文件是一个字节流或二进制流。它把数据看成是一连串的字符（字节）。在 C 语言中对文件的存取是以字符（字节）为单位的，把这种文件称为流式文件。C 语言允许对文件存取一个字符，增加了处理的灵活性。

16.1.3　读文件和写文件

读文件指将磁盘文件中的数据传送到计算机内存中的操作。

写文件指将计算机内存向磁盘文件中传送数据的操作。

16.2　文件指针

在文件的操作中，最关键的因素是“文件指针”。这是因为每个被使用的文件都在内存中开辟一个区域，用来存放文件的有关信息，这些信息保存在一个结构体类型的变量中。该结构体类型是由系统定义的，取名为 FILE（注意大写）。在“stdio.h”文件中有以下的类型定义。

```
typedef struct
{ int _fd;/*文件号*/
  int _cleft;/*缓冲区中剩余的字符*/
  int _mode;/*操作模式*/
  char *_nextc;/*下一个字符位置*/
  char *_buff;/*缓冲区位置*/
}FILE
```

我们可以使用该结构体类型来定义文件型指针变量。

格式如下：

FILE *fp;

fp 是一个指向 FILE 类型结构体的指针变量，可以使 fp 指向某一个文件的结构体变量，从而通过该结构体变量中的文件信息能够访问该文件。也就是说，通过文件指针变量能够找到与它相关的文件，以实现对文件的访问。

16.3 文件的打开与关闭

和其他高级语言一样，对文件读写之前应该“打开”该文件，在使用结束后应该“关闭”该文件。

16.3.1 文件的打开

ANSI C 规定了标准输入输出函数库，用 fopen()函数来实现打开文件。

fopen 函数的调用方式如下。

FILE *fp;

fp=fopen（文件名，使用文件方式）;

【考生注意】

（1）文件名表示准备访问的文件的名称。

（2）打开文件的方式见表 16-2。

表 16-2 文件打开方式

文件使用方式	含义
“r”（只读）	为输入打开一个文本文件
“w”（只写）	为输出打开一个文本文件
“a”（追加）	向文本文件尾部增加数据
“rb”（只读）	为输入打开一个二进制文件
“wb”（只写）	为输出打开一个二进制文件
“ab”（追加）	向二进制文件尾部增加数据
“r＋”（读写）	为读/写打开一个文本文件
“w＋”（读写）	为读/写建立一个新的文本文件
“a+”（读写）	为读/写打开一个文本文件
“rb＋”（读写）	为读/写打开一个二进制文件
“wb＋”（读写）	为读/写建立一个新的二进制文件
“ab＋”（读写）	为读/写打开一个二进制文件

【考生注意】

（1）用“r”方式打开的文件只能用于向计算机输入而不能用做向该文件输出数据，而且该文件应该已经存在，不能打开一个并不存在的用于“r”方式的文件，否则出错。

（2）用“w”方式打开的文件只能用于向该文件写数据，而不能用来向计算机输入。如果原来不存在该文件，则在打开的时候建立一个以指定名字命名的文件。如果原来已经存在一个以该文件命名的文件，则在打开时将该文件删除，重新建立一个新的文件。

（3）如果希望向文件末尾添加新的数据（不希望删除原来的数据），则应该用“a”方式打开。但此时该文件必须已存在，否则将得到出错信息。打开时，位置指针移动到文件末尾。

（4）用“r+”、“w+”、“a+”方式打开的文件可以用来输入和输出数据；用“r+”方式时该文件应该存在，以便能向计算机中输入数据；用“w+”方式则新建立一个文件，先向此文件中写数据，然后可以读此文件中的数据；用“a+”方式打开的文件，原来的文件不被删除，位置指针移动到文件的末尾，可以添加也可以读。

【例题 1】fopen 函数举例。

```
# include "stdio.h"
void main()
{  FILE *fp;/*定义指针变量 fp*/
   if((fp=fopen("c:\\file1.txt","r"))==NULL)/*以只读方式打开文件*/
```

```
    {printf("cannot open this file1.txt\n");
     exit(0);
    }
    ……/*执行其他操作，代码省略*/
}
```

【考生注意】

（1）fopen 的("c:\\file1.txt")表示 C 盘上的 file1.txt 文件，特别注意盘符是 c:\\，而不是 c:\，这是因为在字符串中\\的功能是一个\。

（2）如果 fopen()函数的值为 NULL，即 fp 的值为 NULL，就表明文件可能不存在或路径不对，接着执行 exit(0)语句，exit(0)是系统标准函数，关闭所有打开的文件，并终止程序的执行。

（3）这段程序代码在以后的文件中会经常出现。

16.3.2　文件的关闭

在使用完一个文件后，应该关闭它，以防止它可能再次被误用，“关闭”就是使文件指针变量不指向该文件，与该文件分离。

格式：

fclose（文件指针）

【例题 2】fclose 函数举例。

```
#include  "stdio.h"
void main()
{ FILE *fp;/ *定义指针变量 fp*/
  if((fp=fopen("c:\\file1.txt","r"))==NULL)
    {printf("cannot open this file1.txt\n");
     exit(0);
      }
  ……/*执行其他操作，代码省略*/
  fclose（fp）; /*执行关闭文件的操作*/
}
```

16.4　常用文件的读写操作库函数

文件打开后，就可以对它进行读写操作了，常用的读写函数如下。

16.4.1　格式化读函数和写函数

1. 格式化读 fprintf 函数

格式：

fprintf（文件指针，格式字符串，输出列表）

【作用】

fprintf（文件指针，格式字符串，输出列表）表示将输出列表按格式字符串的形式输出到文件指针指定的文件。

【考生注意】

格式字符串中格式、输出列表都是以逗号分开的。

【例题 3】fprinf 函数举例。

```
#include "stdio.h"
void main()
{ FILE *fp;
  int m=4;
  float n=3.4;
  if((fp=fopen("c:\\f1.txt","w"))==NULL)/*向文件 f1 中写内容，所以采用“w”方式*/
  { printf("can not open the f1!\n");
    exit(0);
  }
  fprintf(fp,"%d,%f",m,n);/*文件 f1 出现有两个值：4 和 3.4*/
  fclose(fp);
}
```

2. 格式化写函数 fscanf 函数

fscanf 函数

格式：

fscanf（文件指针，格式字符串，输入列表）

【作用】

fscanf（文件指针，格式字符串，输入列表）表示将文件指针对应的文件中的内容按照格式字符串的形式输入到输入列表中。

格式字符串中格式、输入列表都是以逗号分开的。

【例题 4】fscanf 函数举例。（假设 C 盘下的 f2 文件中有两个值 3 和 4.5）。

```
#include  "stdio.h"
void main()
{ FILE *fp;
  if((fp=fopen("c:\\f2.txt","r"))==NULL)/*从文件 f2 中读内容，所以采用“r”方式*/
    { printf("can not open the f1!\n");
      exit(0);
    }
  fscanf(fp,"%d,%f",&m,&n);/*把文件中的值赋值给变量 m 和变量 n*/
  printf("%d,%f",m,n);/*输出 m 的值为 3，n 的值为 4.5*/
  fclose(fp);
}
```

16.4.2　读写字符函数 fputc 函数和 fgetc 函数

1. 读字符函数 fputc

格式：

fputc(ch,fp)

【作用】

fputc(ch,fp)的作用是将字符（ch 的值）输出到 fp 所指向的文件。

【考生注意】

ch 是要输出的字符，它可以是一个字符常量，也可以是一个字符变量。

【例题 5】fputc 函数举例——向文件中输入字符以“#”结束。

```
#include  "stdio.h"
void main()
{ FILE *fp
  char  ch;
  if((fp=fopen("c:\\f3.txt","w"))==NULL)/*向文件中输入字符，所以采用“w”方式*/
    {printf("can not open f3!\n");
     exit(0);
    }
  ch=getchar();
  while(ch!='#')/*输入的字符以“#”结束*/
     {fputc(ch,fp);
      ch=getchar();
     }
  fclose(fp);
}
```

2. 写字符函数 fgetc

格式：

ch=fgetc(fp)

【作用】

从文件指针对应的文件中读入一个字符赋值给 ch。

【考生注意】

（1）读入字符结束的标志是 EOF。

（2）在文件内部有一个位置指针，用来指向文件的当前读写字节。在文件打开时，该指针总是指向文件的第一个字节。使用 fgetc 函数后，该位置指针将向下移动一个字节，因此可连续多次使用 fgetc 函数，读取多个字符。应注意文件指针和文件内部的位置指针不是同一个概念。文件指针是指向整个文件的，须在程序中定义说明，只要不重新赋值，文件指针的值就是不变的。文件内部的位置指针用以指示文件内部的当前读写位置，每读写一次，该指针均向后移动，它不需要在程序中定义说明，而是由系统自动设置的。

【例题 6】fgetc 函数举例。

```
#include  "stdio.h"
void main()
{ FILE *fp
  char  ch;
  if((fp=fopen("c:\\f4.txt","r"))==NULL)/*向文件中读出字符，所以采用“r”方式*/
    {printf("can not open f4!\n");
     exit(0);
  }
  while((ch=fgetc(fp))!=EOF)/*从文件中读出字符，以 EOF 结束*/
    printf("%c",ch);/*输出字符*/
  fclose(fp);
}
```

16.4.3 读写字符串函数 fgets 函数和 fputs 函数

与前面所学的输入输出字符串函数一样，文件也有读写字符串函数。

1. 读字符串函数 fgets

格式：

fgets(str,n,fp)

【作用】

从 fp 指向的文件读出 n-1 个字符，并把它们存放到字符数组 str 中，如果在读入 n-1 个字符结束之前遇到换行符或 EOF，读入即结束。字符串读入会在最后加一个‘\0’字符，fgets 函数返回值为 str 的首地址。

【考生注意】

fgets()函数返回的是一个地址，而不是一个值。

【例题 7】fgets 函数举例——文件 f5 中有“abcdefg”7 个字符。

```
#include  "stdio.h"
void main()
{ FILE *fp;
  char *p;
  char str[10]
  if((fp=fopen("c:\\f5.txt","r"))==NULL)/*向文件中读出字符，所以采用“r”方式*/
    {printf("can not open f5!\n");
     exit(0);
    }
  p=fgets(str,8,fp);/*从 fp 对应的文件 f5 中读入 7（8-1）个字符到数组 str 中，并把数组的首地址赋值给指针变量 p*/
  printf("%s",p);/*输出指针变量 p 所对应的内容——abcdefg*/
  printf("%s",str); /*输出数组 str 的内容——abcdefg*/
  fclose(fp);
}
```

2. 写字符串函数 fputs

格式：

fputs(str,fp)

【作用】

将字符串 str 写入到 fp 所对应的文件中。

【考生注意】

fputs()函数中的第一个参数可以是字符串常量、字符数组名或字符型指针。

【例题 8】fputs 函数举例。

```
#include  "stdio.h"
void main()
{ FILE *fp;
  char str[10]＝"abcdefg";
  if((fp=fopen("c:\\f6.txt","w"))==NULL)/*向文件中写入字符，所以采用“w”方式*/
    {printf("can not open f6!\n");
     exit(0);
    }
  fputs(str,fp);/*向指针 fp 所对应的文件中写入字符串的内容*/
```

```
    fclose(fp);
}
```

16.4.4　读写数据块函数 fread 和 fwrite 函数

fgetc()和 fputc()函数一次只能读/写一个字节数据，但在实际的应用过程中，却要求常常读/写 1 个数据块（若干个字节）

1. 读数据块函数 fread

格式：

fread(buffer,size,count,fp)

其中，buffer：是一个指针，它是读入数据的存放地址。

size：要读的字节数。

count：要进行读多少个 size 字节的数据项。

fp：文件类型指针。

【作用】

从 fp 所指向的文件的当前位置开始，一次性读入 size 个字节，重复 count 次，并将读入的数据存放到从 buffer 开始的内存中，同时将指针向前移动 size*count 个字节。

2. 写数据块函数 fwrite

格式：

fwrite(buffer,size,count,fp)

其中，buffer：是一个指针，它是写入数据的存放地址。

size：要写的字节数。

count：要进行写多少个 size 字节的数据项。

fp：文件类型指针。

【作用】

从 buffer 开始的内存中，一次性输出 size 个字节，重复 count 次，并将写入的数据存放到 fp 所指向的文件中，同时将指针向前移动 size*count 个字节。

16.5　文件定位函数

前面介绍了对文件的读写方式是顺序读写，即读写文件只能从头开始，顺序读写各个数据。但有时只需要读写文件中某一指定的部分。此时可移动文件内部的位置指针到需要读写的位置上，再进行读写，这种读写操作称为随机读写。

实现随机读写的关键是要按要求移动位置指针，这称为文件的定位。

移动文件内部位置指针的函数主要有两个，即 fseek 函数和 rewind 函数。另外，函数 ftell 用来得到文件指针的当前位置，用相对于文件头的字节位移量表示。

16.5.1　fseek 函数

格式：int fseek(FILE*fp,long offset,int from)

功能：移动文件位置指针到指定位置。

说明：

（1）fseek()把文件位置指针移动到与 from 所指定的文件位置距离 offset 字节处，如果指针移动成功，则返回 0；出错，则返回非 0。

（2）参数 offset 为字节偏移量，为长整型数据，正数代表前进，负数代表后退。

（3）参数 from 代表移动的开始位置，其值可以是 0，1 或 2 中的一个，分别代表文件开始位置 SEEK_SET(0)，文件

当前位置 SEEK_CUR(1)和文件末尾 SEEK_END(2)。SEEK_SET，SEEK_CUR 和 SEEK_END 这 3 个符号常量定义在头文件“stdio.h”中。

假设 fp 已指向一个二进制文件，以下函数调用可使文件位置指针从文件的开头后移 30 个字节。

seek(fp,30L,SEEK_SET)

若 fp 已指向一个二进制文件，以下函数调用可使文件位置指针从文件尾部前移 10 sizeof（int）即 20 个字节。

fseek(fp,-10L*sizeof (int),SEEK_END)

对于文本文件，位移量必须是 0。假设 fp 已指向一个文本文件，以下函数调用可使文件位置指针移到文件的开始。

seek(fp,0L,SEEK_SET)

假设 fp 已指向一个文本文件，以下函数调用使文件位置指针移到文件的末尾。

fseek(fp,0L,SEEK_END)

16.5.2 ftell 函数

格式：long ftell(FILE *fp)

功能：得到 fp 指向的文件的指针位置。

说明：ftell()在调用成功后返回当前指针位置，出错时返回-1L。

当打开一个文件时，通常并不知道该文件的长度，通过以下函数调用可以求出文件的字节数。

```
fseek(fp,0L,SEEK_END);   /*把位置指针移到文件末尾*/
t=ftell(fp);
```

若二进制文件中存放的是 struct st 结构体类型数据，则通过以下语句可以求出该文件中以该结构体为单位的数据块的个数。

```
fessk(fp,0L,SEEK_END);
t=ftell(fp);
n=t/sizeof(struct st);
```

16.5.3 rewind 函数

格式：void rewind(FILE *fp)

功能：将文件位置指针定位于文件的开始处。

说明：该函数的作用是把文件位置指针返回到文件开始处，清除文件结束标志和出错标志，无返回值。

16.6 习题

16.6.1 选择题

1. 使用 fgets(str,n,fp)函数从文件中读入一个字符串，下列叙述中错误的是（　　）。

A）字符串读入后会自动加入‘\0’　　B）fp 是指向该文件的文件型指针

C）fgets 函数将从文件中最多读入 n-1 个字符　　D）fgets 函数将从文件中最多读入 n 字符

2. 函数 ftell(fp)的作用是（　　）。

A）得到 fp 所指向文件的当前读写位置　　B）初始化流式文件的位置指针

C）移动流式文件的位置指针　　D）以上答案均正确

3. 已知函数的调用形式为“fread(buffer,size,count,fp);”，则其中 buffer 代表的是（　　）。

A）一个整型变量，代表要读入的数据项总数　　B）一个文件指针，指向要读的文件

C）一个指针，是指向的输入数据放在内存中的起始位置　　D）一个存储区，存放要读的数据项

4. 有以下程序：

```
#include  <stdio.h>
void writestr(char *fn,char *str)
{ FILE *fp;
  fp=fopen(fn,"w");fputs(str,fp);fclose(fp);}
 main()
{ writestr("t1.dat","start");
  writestr("t1.dat","end");}
```

程序运行后，文件 t1.dat 中的内容是（　　）。

A）start　　B）end　　C）startend　　D）endrt

5. 以下程序的功能是（　　）。

```
#include  <stdio.h>
main()
{
    FILE *fp;
    char str[ ]="HELLO";
    fp=fopen("PRN","w");
    fputs(str,fp);
    fclose(fp);
}
```

A）在屏幕上显示“HELLO”　　B）把“HELLO”存入 PRN 文件中

C）在打印机上打印出“HELLO”　　D）以上都不对

6. 下列叙述中正确的是（　　）。

A）C 语言中的文件是流式文件，因此只能顺序存取数据

B）打开一个已存在的文件并进行了写操作后，原有文件中的全部数据必定被覆盖

C）在一个程序中当对文件进行了写操作后，必须先关闭该文件然后再打开，才能读到第 1 个数据

D）当对文件的读（写）操作完成之后，必须将它关闭，否则可能导致数据丢失

7. 有以下程序：

```
#include  <stdio.h>
main()
{
   FILE *fp;int k,n,a[6]={1,2,3,4,5,6};
   fp=fopen("d2.dat","w");
   fprintf(fp,"%d%d%d\n",a[0],a[1],a[2]);
   fprintf(fp,"%d%d%d\n",a[3],a[4],a[5]);
   fclose(fp);
   fp=fopen("d2.dat","r");
   fscanf(fp,"%d%d",&k,&n);printf("%d%d\n", k,n);
   fclose(fp);
}
```

程序运行后的输出结果是（　　）。

A）12　　B）14　　C）1234　　D）123　456

8. 在“文件包含”预处理语句的使用形式中，当#include 后面的文件名用“”（双引号）括起时，寻找被包含文件的方式是（　　）。

A）直接按系统设定的标准方式搜索目录

B）先在源程序所在的目录搜索，如没找到，再按系统设定的标准方式搜索

C）仅仅搜索源程序所在目录

D）仅仅搜索当前目录

9. 函数 ftell(fp)的作用是（　　）。

A）得到 fp 所指向文件的当前读写位置　　B）初始化流式文件的位置指针

C）移动流式文件的位置指针　　D）以上答案均正确

10. 若 fp 是指向某文件的指针，且尚未读到文件末尾，则函数 feof(fp)的返回值是（　　）。

A）EOF　　B）-1　　C）非零值　　D）0

11. seek 函数的正确调用形式是（　　）。

A）fseek（位移量，起始点，文件类型指针）　　B）fseek（文件类型指针，位移量，起始点）

C）fseek（文件类型指针，起始点，位移量）　　D）fseek（起始点，位移量，文件类型指针）

16.6.2　填空题

1. 设有定义“FILE *fw;”，请将以下打开文件的语句补充完整，以便可以向文本文件 readme.txt 的最后续写内容。

```
fw=fopen("readme.txt",_________);
```

2. 下述程序的功能是：以二进制“写”方式打开文件 d1.dat，写入 1～100 这 100 个整数后关闭文件，再以二进制“读”方式打开文件 d1.dat，将这 100 个整数读入到另一个数组 b 中，并打印输出。请填空。

```
#include  <stdio.h>
main()
{
   FILE*fp;
   int i,a[100],b[100];
   fp=fopen("d1.dat","wb");
   for(i=0;i<100;i++),a[i]=i+1;
      fwrite(a,sizeof(int),100,fp);
   fclose(fp);
   fp=fopen("d1.dat",__________);
   fread(b,sizeof(int),100,fp);
   fclose(fp);
   for(i=0;i<100;i++)
      frintf("%d\n",b[i]);
}
```

16.6.3　程序设计题

1. 从键盘上输入一个字符串，要求将其中的大写字母转换成小写字母，然后把整个字符串存储到磁盘文件中。

2. 将字符串 s1 中的内容“abcdefg”和字符串 s2 中的“12345”变成磁盘文件 C 中的内容“12345abc”。

第17章 上机指导

17.1 上机应试技巧

一、上机环境介绍

Visual C++ 6.0 IDE 是目前计算机二级考试指定的 C 语言上机考试环境。我们首先来熟悉一下 Visual C++的环境。

（1）打开 Microsoft Visual C++的界面如图 17-1 所示。

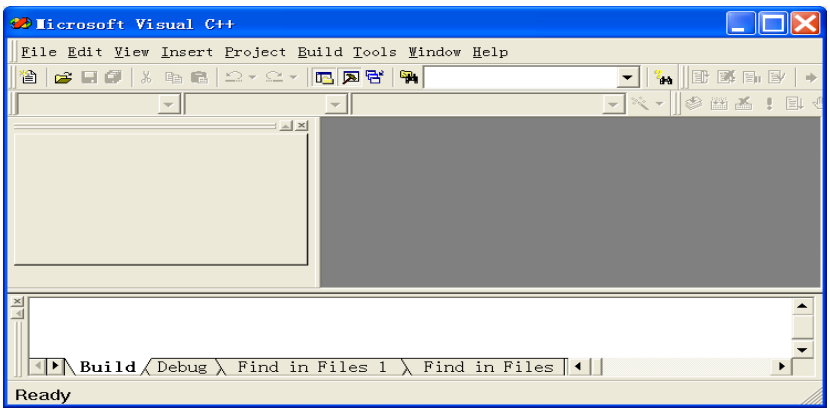


图 17-1　Visual C++界面

（2）在“File”菜单下选择“New”命令，出现如图 17-2 所示的对话框。

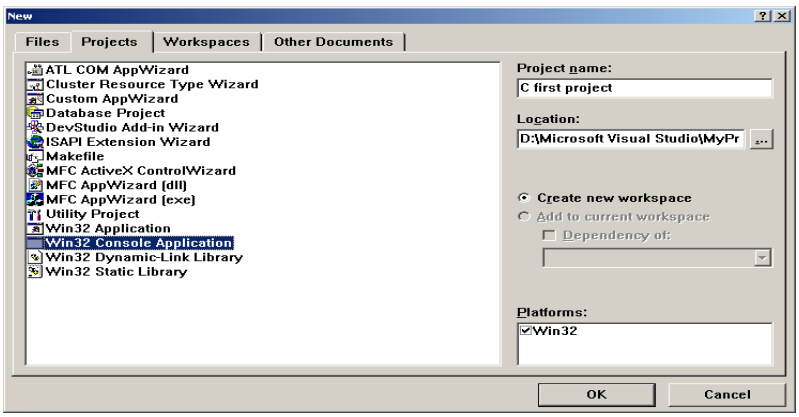


图 17-2 “New”对话框

这里，可以看到 VC 新建的类型有“file”，“project”，“workspace”，“other documents”，现在默认创建一个工程（project），在左边窗口中可以看到 VC 所能创建的工程类型。我们所选择的“Win32 Console Application”类型是 Win32 的控制台应用程序类型。在“Project name”文本框中输入要创建的工程名称，在“Location”文本框中添入工程保存的路径，也可以通过右边的按扭进行路径选择。其他参数保持默认即可，单击“OK”按钮。

（3）单击“OK”按钮之后出现如图 17-3 所示的界面。

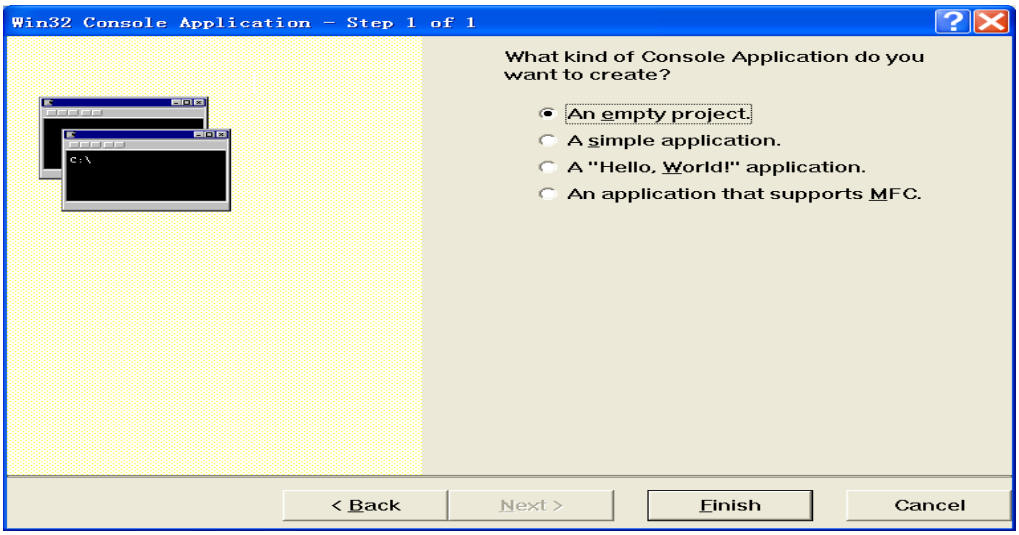


图 17-3 “Win32 Console Application-Step 1 of 1”对话框

这里，出现的界面是 Win32 控制台应用程序所能创建的类型，我们调试单文件程序可以选中第一个单选项，如果是多文件，即含有自己编写的头文件（.h）和多个源文件（.cpp）的工程，我们可以选中第二个单选项。现在我们选中第一个单选项，单击“Finish”按钮继续。

（4）之后是一个简单的对工程的说明对话框，至此，我们创建了一个空的工程“An empty project”（上一步我们选择的）。这个工程没有任何文件，没有任何内容，比如一个空的房子，没有任何家具，不能从事任何活动。所以我们要添加工程内容（文件）。选择“File”菜单下的“New”命令，打开默认的是新建文件面板，如图 17-4 所示。

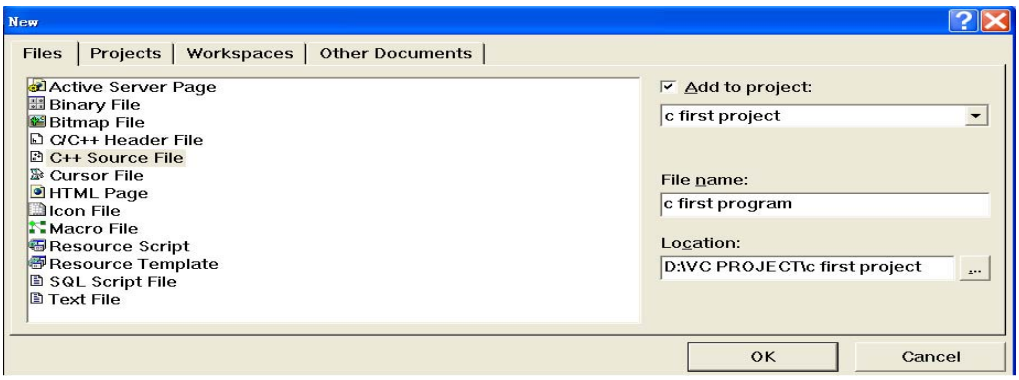


图 17-4 “File”选项卡

这里，可以看到左边窗口中的是 VC 所创建的文件类型，我们选择“C++ source file”选项，在“File name”文本框中输入要创建文件的名称，其他默认，继续单击“OK”按钮。VC IDE 变成如图 17-5 所示的界面。

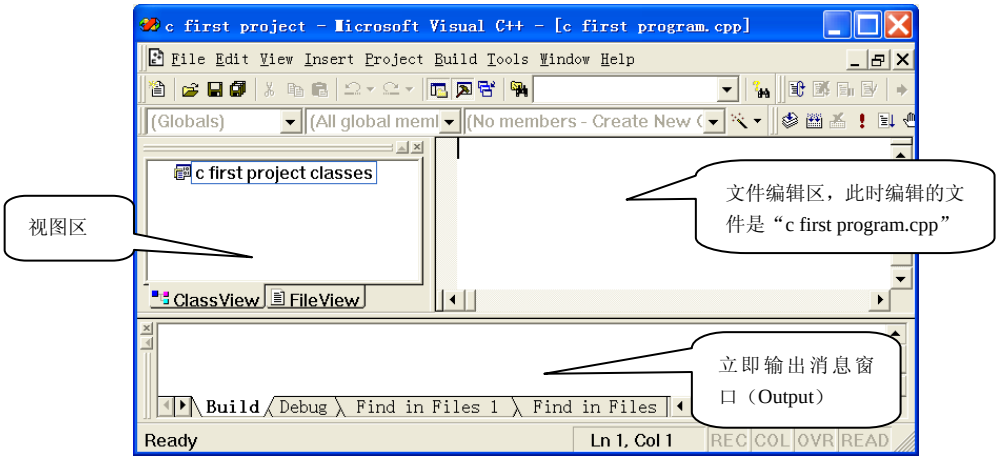


图 17-5　IDE 新界面

应用程序的运行。

（1）在上图中的文件编辑区内输入代码清单中的程序，这个过程也就是程序的编辑过程。

（2）我们先看几个常用的工具图标，如图 17-6 所示。

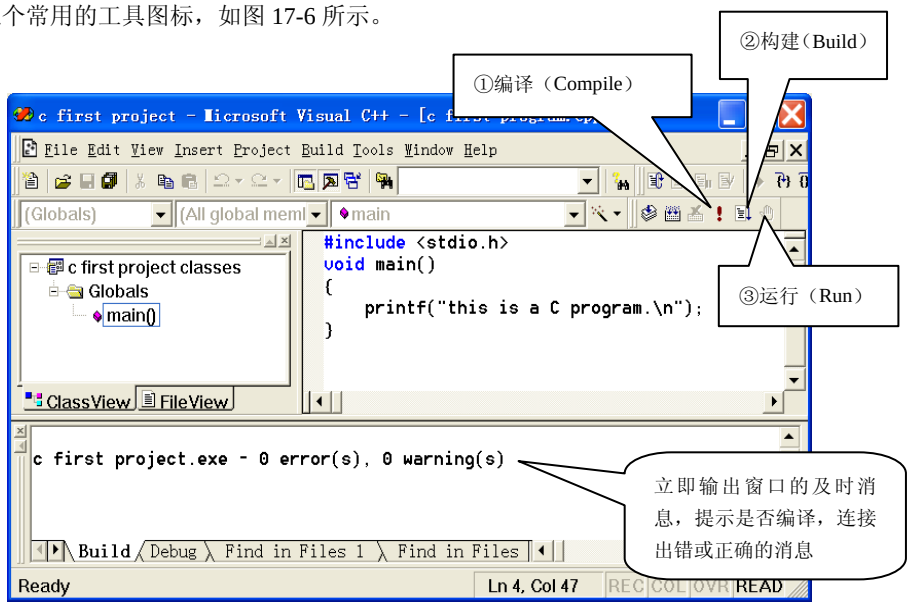


图 17-6　常用工具图标

这里，源文件编辑好，然后①编译，如果有语法错误在立即输出窗口有提示信息，可以修改源程序；②构建，编译提示“c first program。obi － 0 error(s)，0 warning(s)”表示源文件的目标文件生成；③运行，生成 exe 文件执行。可能有不能连接的情况，出错消息也会在立即输出窗口中显示，根据情况调试。

（3）正确运行的结果如图 17-7 所示。

图 17-7　运行结果

二、上机操作流程

（1）候考：进入候考室，按监考老师的安排进入机房。

（2）登录：在考试系统上输入准考证号，核对身份。

（3）抽题：如遇机器死机、无法抽题等异常情况，应迅速报告监考老师。

（4）答题：注意一定要将做题结果保存在考生文件夹下，否则做题无效。

（5）交卷：单击“考试界面”上方的考试信息窗口中的“交卷”按钮。注意一旦交卷就无法更改。

（6）评分：部分考点在交卷后一段时间内由监考老师通过局域网收题，当场可以评分，等笔试成绩出来后，一起公布。

三、上机考试纪律

（1）上机考试时，考生应该在规定的考试时间提前 30 分钟报到，交验准考证和身份证（军人身份证或户口本），同时抽签决定上机考试的工作站号（或微机号）。

（2）考生提前 5 分钟进入机房，坐在由抽签决定的工作站号（或微机号）上，不允许乱坐位置。

（3）不得擅自登录与自己无关的考号。

（4）不得擅自复制或删除与自己无关的目录和文件。

（5）考生不得在考场中交头接耳、大声喧哗。

（6）未到 10 分钟不得离开考场。

（7）迟到 10 分钟取消考试资格。

（8）考试中计算机出现故障、死机、死循环、电源故障等异常情况（即无法进行正常考试时），应举手示意与监考人员联系，不得擅自关机。

（9）考生答题完毕后应立即离开考场，不得干扰其他考生答题。

注意：考生必须在自己的考生目录下进行考试，否则在评分时会查询不到考试内容而影响考试成绩。

四、操作步骤及考试原则

1. 考试时间

（1）二级 C 语言上机考试时间定为 90 分钟。考试时间由上机考试系统自动进行计时，提前 5 分钟自动报警来提醒考生按时存盘。考试时间用完，上机考试系统将自动锁定计算机，考生将不能继续进行考试。

（2）当考生登录成功后，系统将自动抽取考题并且在屏幕上显示上机须知，提示考生按“S”键开始考试，系统开始计时；如果是第二次登录，则系统将累计计时，考生必须在规定的时间内完成考试内容。当考生超出考试所用时间时机器将自动关闭；当考试只剩下指定时间时，屏幕上会自动报告所剩考试时间，此时考生只需按任意键继续答题，不会影响考生成绩。

2. 考题类型及分值

目前二级 C 语言考试题类型有 3 种：基本操作题、简单应用题和综合应用题，满分为 100 分。

3. 上机考试步骤

考试过程分为登录、查题、答题、交卷等几个阶段。

（1）登录

根据考试要求打开考试系统软件，启动考试程序，出现如图 17-8 所示的登录界面。

图 17-8　登录界面

在实际答题之前，需要先进行考试系统登录。一方面，这是考生姓名的记录凭据，系统需要验证磁盘中考生姓名和考号是否与本人相符；另一方面，考试系统也需要对每一位考生随机选择一套试题，生成试卷。单击图 17-8 中的“开始登录”按钮或按回车键，出现考号输入窗口，如图 17-9 所示。

在图 17-9 中输入正确的准考证号、考生姓名、身份证号，再单击考号验证按钮按回车键对输入的考号、姓名和身份证号进行验证，出现如图 17-10 所示的界面。

在确定是正确的信息后，单击“是”按钮，进入到考生的准备答题界面，如图 17-11 所示。

单击“开始答题并计时”按钮后，进入到试题界面，如图 17-12 所示。

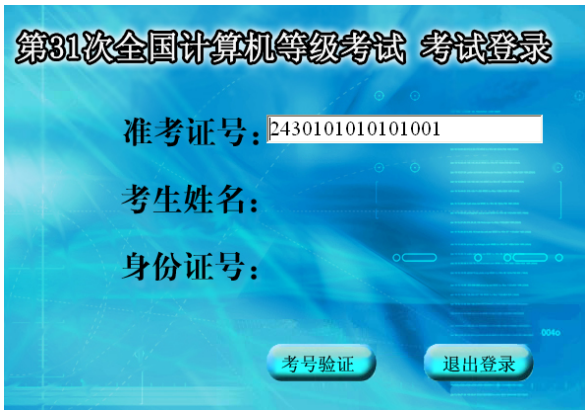


图 17-9　考号验证

图 17-10　确认考生信息

图 17-11　准备答题

图 17-12 试题界面

（2）查题

选择图 17-12 中的“答题”菜单下的“启动 Visual C++”命令，进入到查题界面中，如图 17-13 所示。

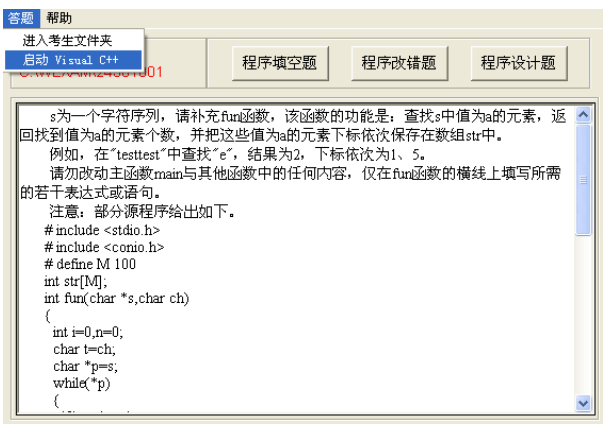


图 17-13　启动 Visual C++界面

（3）答题

在图 17-13 的界面中，选择“文件”菜单下的“打开”命令，进入代码界面，如图 17-14 所示。

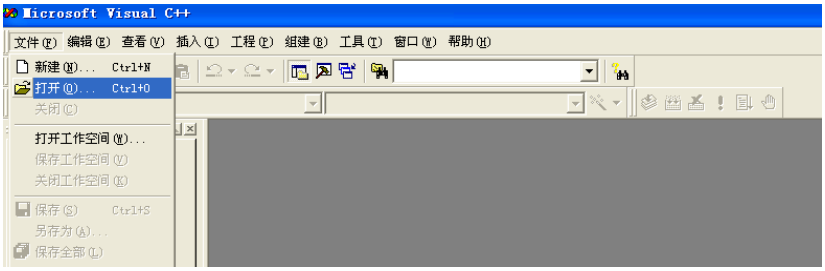


图 17-14　试题启动界面

（4）交卷

在图 17-15 中，进行交卷的确认，单击“确定”按钮，进行交卷。

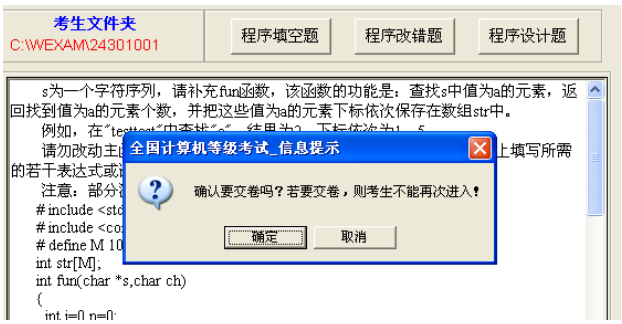


图 17-15　交卷界面

五、上机做题时应注意的事项

上机考试的评分是以机评为主，人工复查为辅的。机评当然不存在公正性的问题，但却存在呆板的问题，有时还可能因为出题者考虑不周出现错评的情况。考生做题时不充分考虑到这些情况，也可能吃亏。

1. 改错题的特点和注意事项

（1）上机改错试题中通常包含 2 个（或 3 个）错误需要修改。

（2）试题中用“/******found******/”来提示在下一行（或下面第 2 行）有错。

（3）错误的性质基本分语法错误和逻辑错误两种，也有些试题要求把语句添加在下画线处。

（4）特别要注意的是：只能在出错的行上进行修改，不要改动程序行的顺序，更不要自己另编程序。

2. 上机改错测试时，建议按以下步骤进行

（1）首先仔细审题，了解试题的要求，看清楚试题给出的输入和输出示例，以便检验改错后程序运行的结果是否正确。

（2）当在 Visual C++环境下调出源程序后，审视“/******found******/”所在行，根据题义理解程序所采用的基本算法，做到心里有数。

（3）当编译提示有语法错误时，可参考编译提示来查找并改正错误。

（4）当不再出现语法错误时，按照试题的示例给出的数据进行试算，若试算的结果与给出的输出结果相同时，该题就做对了；若试算的结果与给出的输出结果不同，应进一步检查程序中的逻辑错误。

（5）当程序存在逻辑错误时，首先应当理解题意，读懂程序的算法，必要时可按步检查数据的流程，以便确定错误所在。例如，题目要求数据按由小到大的顺序排序，而结果数据按由大到小进行了排序，问题可能出现在条件判断上。又如，输出的字符串比预期的短，就有可能是字符串的结束标志放错了位置。再如做循环的时候，数组上限下限错误了，基数是从 0 开始而不是从 1 开始的。修改程序中的逻辑错误时，要求考生认真读懂程序代码。

（6）修改完成，得到正确结果后，一定不要忘记把修改后的程序存盘。

3. 上机编程题的特点和注意事项

在二级 C 程序设计上机考试中，要求完成一个独立的函数的编程。目前教育部考试中心已出版了上机考试习题集，这些有助于学习编程，但考生应当在学习过程中理解基本的算法，通过实际的上机操作积累经验，才能掌握基本的编程能力。

进行编程测试时，建议按以下步骤进行。

（1）首先仔细审题，了解试题的要求，记下试题给出的输入和输出示例，以便检验在完成指定函数后程序运行的结果是否正确。

（2）当在 Visual C++环境下调出源程序后，应对照函数首部的形参，审视主函数中调用函数时的实参内容，以便明

确函数中需要处理的数据对象。

（3）理解试题的要求，审视主函数中调用函数的方式，若在表达式中调用函数（如把函数值赋给某个对象），则要求有函数值返回，需注意函数的类型，并在函数中用 return 语句返回函数值；若主函数中仅用语句形式调用函数，则需要通过形参间接地返回所得结果。

（4）选择适当的算法进行编程，输入程序语句，不要忘记及时存盘！

（5）编译程序，直到没有语法错误。

（6）调试程序，利用试题中给出的示例数据进行输入（若要求输入的话），运行程序，用示例的输出数据检验输出结果，直到结果相同。

17.2　上机模拟试题一

17.2.1　程序填空题

s 为一个字符序列，请补充 fun 函数，该函数的功能是：查找 s 中值为 a 的元素，返回找到值为 a 的元素的个数，并把这些值为 a 的元素的下标依次保存在数组 str 中。

例如，在“testtest”中查找“e”，结果为 2，下标依次为 1，5。

请勿改动主函数 main 与其他函数中的任何内容，仅在 fun 函数的横线上填写所需的若干表达式或语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
#include <conio.h>
#define M 100
int str[M];
int fun(char *s,char ch)
{
  int i=0,n=0;
  char t=ch;
  char *p=s;
  while(*p)
  {
    if(___1___)
      ___2___;
    p++;
    i++;
  }
  return ___3___;
}
main()
{
  char s[M];
  char ch;
  int i,n;
  printf("Please Input the original string\n ");
  gets(s);
  printf("The Original string is :\n");
  puts(s);
```

```
  printf("Input character\n");
  scanf("%c",&ch);
  n=fun(s,ch);
  printf(" \nThe number of character is: %d\n",n);
  printf("The position of character:\n");
  for(i=0;i<n;i++)
     printf(" %d ",str[i]);
}
```

17.2.2 程序改错题

下列给定程序中，函数 fun 的功能是：用递归算法计算斐波拉契级数数列中第 n 项的值。从第 1 项起，斐波拉契级数序列为 1，1，2，3，5，8，……例如，若为 n 输入 7，则该项的斐波拉契级数值为 13。

请修改程序中的错误，得出正确的结果。

注意：不要改动 main 函数，不能增行或删行，也不能更改程序的结构。

```
#include <stdio.h>
long fun(int m)
{
  /*******found********/
  switch(m);
  {
    case 0:
       return0;
    /********found********/
    case 1:
    case 2:
       return 1;
  }
  return (fun(m-1)+fun(m-2));
}
main()
{
  long a;
  int n;
  printf("Input n:  ");
  scanf("%d",&n);
  printf("n=%d\n",n);
  a=fun(n);
  printf("a=%d\n\n",a);
}
```

17.2.3 程序设计题

请编写函数fun，该函数的功能是：删除一维数组中所有相同的数，使之只剩一个。数组中的数已按由小到大的顺序排列，函数返回删除后数组中数据的个数。

例如，若一维数组中的数据是：

1,1,1,2,2,3,3,3,3,4

删除后，数组中的内容应该是：

1,2,3,4

请勿改动主函数 main 与其他函数中的任何内容，仅在函数 fun 的花括号中填入所编写的若干语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
#define M 100
int fun(int b[],int m)
{

}
main()
{
  int b[M]={1,1,1,2,2,3,3,3,3,4},i,m=10;
  FILE *out;
  printf("The original data :\n");
  for(i=0;i<m;i++)
    printf("%3d",b[i]);
  m=fun(b,m);
  printf("\nThe data after deleted :\n");
  out=fopen ("outfile.dat","w");
  for(i=0;i<m;i++)
  {
    printf("%3d",b[i]);
    fprintf(out,"%d\n",b[i]);
  }
  fclose(out);
}
```

17.3　上机模拟试题二

17.3.1　程序填空题

给定程序中，函数 fun 的功能：将 s 所指字符串中的所有非数字字符移到所有数字字符之后，并保持数字字符串和非数字字符串原有的先后次序。

例如，形参 s 所指的字符串为：asf34fgrt5657gngjh78。则执行结果为：34565778asffgrtgngjh。

请勿改动主函数 main 与其他函数中的任何内容，仅在 fun 函数的横线上填写所需的若干表达式或语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
void fun(char *str)
{
  int i,j=0,k=0;
  char temp1[80],temp2[80];
  for(i=0;str[i]!='\0';i++)
    if(str[i]>='a' && str[i]<='z')
```

```
    {
      ___1___;
      j++;
    }
    else
    {
      temp1[k++]=str[i];
    }
  temp2[j]=0;
  temp1[k]=0;
  for(i=0;i<k;i++)
    ___2___;
  for(i=0;i<___3___;i++)
    str[k+i]=temp2[i];
}
main()
{
  char str[100]="asf34fgrt5657gngjh78";
  printf("\nThe original string is : %s\n",str);
  fun(str);
  printf("\nThe result is : %s\n",str);
}
```

17.3.2 程序改错题

下列给定程序中，函数 fun 的功能是：计算并输出 max 以内最大的 10 个素数之和。最大值由主函数传给 fun 函数。若 max 的值为 50，则函数的值为 300。

请修改程序中的错误，使程序能得出正确的结果。

注意：不要改动 main 函数，不能增行或删行，也不能更改程序的结构。

```
#include <conio.h>
#include <stdio.h>
#include <math.h>
int fun(int max)
{
  int sum=0,n=0,j,flag;
  while((max>=2)&&(n<10))
  {
    flag=1;
    for(j=2;j<=max/2;j++)
      /********found********/
      if(max%j==0)
      {
        flag=0;
        break
      }
    if(flag)
    {
      sum+=max;
      n++;
```

```
    }
    max--;
  }
  return sum;
}
main()
{
  printf("%d\n",fun(50));
}
```

17.3.3　程序设计题

请编写函数 fun，该函数的功能是：将放在字符串数组中的 M 个字符串（每串的长度不超过 N），按顺序合并组成一个新的字符串。

例如，若字符串数组中的 M 个字符串是：

1111
2222222
33
4444

则合并后的字符串的内容应是 11112222222334444。

请勿改动主函数 main 与其他函数中的任何内容，仅在函数 fun 的花括号中填入所编写的若干语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
#define M 4
#define N 20
void fun(char str[M][N],char *a)
{

}
main()
{
  char matrix[M][N]={"1111","2222222","33","4444"},i;
  char str [100]={"****************"};
  FILE *out;
  printf("The string:\n");
  for(i=0;i<M;i++)
    puts(matrix[i]);
  printf("\n");
  fun(matrix,str);
  printf("The string:\n");
  printf("%s",str);
  printf("\n\n");
  out=fopen("outfile.dat","w");
  fprintf(out,"%s",str);
  fclose(out);
}
```

17.4 上机模拟试题三

17.4.1 程序填空题

请补充函数 fun，该函数的功能是：统计所有小于等于 x（x>2）的素数的个数，素数的个数作为函数值返回。例如，输入 x=20，结果：2,3,5,7,11,13,17,19。

请勿改动主函数 main 与其他函数中的任何内容，仅在 fun 函数的横线上填写所需的若干表达式或语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
int fun(int x)
{
  int i,j,count=0;
  printf("\nThe prime number between 2 to %d\n",x);
  for(i=2;i<=x;i++)
  {
    for(___1___;j<i;j++)
      if(___2___%j==0)
        break;
    if(___3___>=i)
    {
      count++;
      printf(count%15? "%5d" : "\n%5d",i);
    }
  }
  return count;
}
main()
{
  int x=20,result;
  result=fun(x);
  printf("\nThe number of prime is :  %d\n",result);
}
```

17.4.2 程序改错题

下列给定程序中函数 fun 的功能是：计算 m!。例如，给 m 输入 3，则输出 6.000000。

请修改程序中的错误，使程序能输出正确的结果。

注意：不要改动 main 函数，不能增行或删行，也不能更改程序的结构。

```
#include <stdio.h>
#include <conio.h>
double fun(int m)
{
  double result=1.0;
  /********found********/
  if m==0
    return 1.0;
```

```
  while(m>1&&m<170)
    /*******found*********/
    result=m--;
  return result;
}
main()
{
  int m;
  printf("Input m:");
  scanf("%d",&m);
  printf("\n\n%d!=%1f\n\n",m,fun(m));
}
```

17.4.3　程序设计题

学生的记录由学生和成绩组成，M 名学生的数据已在主函数中放入结构体数组 stu 中，请编写函数 fun，它的功能是把分数最高的学生数据放在 high 所指的数组中。注意：分数最高的学生可能不止一个，函数返回分数最高的学生的人数。

请勿改动主函数 main 与其他函数中的任何内容，仅在函数 fun 的花括号中填入所编写的若干语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
#define M 10
typedef struct
{
  char num[10];
  int s;
}  SCORE;
int fun(SCORE *p,SCORE *q)
{

}
main ()
{
  SCORE stu[M]={{"02",69},{"04",85},{"01",91},{"08",64},{"06",87},{"015",85},{"013",91},{"012",
  64},{"011",91},{"017",64}};
  SCORE high[M];
  int i,n;
  FILE *out;
  n=fun(stu,high);
  printf("The %d high score :\n",n);
  for(i=0;i<n;i++)
    printf("%s %4d\n",high[i].num,high[i].s);
  printf("\n");
  out=fopen ("outfile.dat","w");
  fprintf(out,"%d\n",n);
  for(i=0;i<n;i++)
    fprintf(out,"%4d\n",high[i].s);
```

```
    fclose(out);
}
```

17.5 上机模拟试题四

17.5.1 程序填空题

请补充 fun 函数，该函数的功能是求能整除 k 且是偶数的数，把这些数保存在数组 a 中，并按从大到小的顺序输出。例如，当 k=40 时，依次输出 40 20 10 8 4 2。

请勿改动主函数 main 与其他函数中的任何内容，仅在 fun 函数的横线上填写所需的若干表达式或语句。

注意：部分源程序给出如下。

```
#include <conio.h>
#include <stdio.h>
void fun(int k,int a[])
{  int i;
  int j=0;
  for(___1___;i<=k;i++)
    if(k%i==0 ___2___ i%2==0)
      a[j++]=i;
  printf("\n\n ");
  for(i=___3___;i>=0;i--)
    printf("%d ",a[i]);
}
main()
{
  int k=1;
  int a[100];
  printf("\nPlease input k\n");
  scanf("%d",&k);
  fun(k,a);
}
```

17.5.2 程序改错题

下列给定程序的功能是：读入一个英文文本行，将其中每个单词的第一个字母改成大写，然后输出此文本行（这里的“单词”是指由空格隔开的字符串）。例如，若输入“good luck!”，则应输出“Good Luck!”。

请修改程序中的错误，使程序能得出正确的结果。

注意：不要改动 main 函数，不能增行或删行，也不能更改程序的结构。

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
/*******found*********/
void top(char s)
{
  int i=0;
```

```
    for(;*s;s++)
      if(i)
      {
        if(*s==' ')
          i=0;
      }
      else
      {
        if(*s!=' ')
        {
          i=1;
          *s=toupper(*s);
        }
      }
}
main()
{
  char str[81];
  printf("\nPlease enter an English text line: ");
  gets(str);
  printf("\n\nBefore changing:\n %s",str);
  top(str);
  printf("\nAfter changing:\n %s\n",str);
}
```

17.5.3　程序设计题

假定输入的字符串中只包含字母和#号。请编写函数 fun，它的功能是：除了字符串前导和尾部的#号之外，将串中其他的#号全部删除。形参 r 已指向字符串中第一个字母，形参 v 已指向字符串中最后一个字母。在编写函数时，不得使用 C 语言提供的字符串函数。

例如，若字符串中的内容为“####a#bc#def##g#####”，删除后，字符串中的内容则应当是“####abcdefg#####”。

请勿改动主函数 main 与其他函数中的任何内容，仅在函数 fun 的花括号中填入所编写的若干语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void fun(char *s,char *r,char *v)
{

}
main()
{
  char str[100],*p,*q;
  FILE *out;
  printf("Enter a string:\n");
  gets(str);
```

```
  p=q=str;
  while(*p)
    p++;
  p--;
  while(*p=='#')
    p--;
  while(*q=='#')
    q++;
  fun(str,q,p);
  printf("The string after deleted:\n");
  puts(str);
  out=fopen ("outfile.dat","w");
  strcpy(str,"######a#b#c#d######");
  fun(str,str+4,str+13);
  fprintf(out,"%s",str);
  fclose(out);
}
```

17.6 上机模拟试题五

17.6.1 程序填空题

请补充 main 函数，该函数的功能是：输出一个 N×N 矩阵，要求周边元素赋值 0，非周边元素赋值 1。

仅在横线上填写所需的若干表达式或语句，请勿改动函数中的其他任何内容。

注意：部分源程序给出如下。

```
#include <stdio.h>
#define N 10
main()
{
  int a[N][N];
  int i,j,n;
  printf("Please input n:\n");
  scanf("%d",&n);
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    {
      if(i==0||i==n-1||j==0||j==n-1)
        ___1___;
      else
        ___2___;
    }
  printf("The result is:\n");
  for(i=0;i<n;i++)
  {
    printf("\n");
```

```
    for(j=0;j<n;j++)
      printf("%4d",a[i][j]);
  }
}
```

17.6.2　程序改错题

下列给定的程序中，函数 fun 的功能是交换主函数中两个变量的值。例如，若变量 x 中的值为 1，y 中的值为 2，则程序运行后 x 中的值为 2，y 中的值为 1。

请修改程序中的错误，得出正确的结果。

注意：不要改动 main 函数，不能增行或删行，也不能更改程序的结构。

```
#include <stdio.h>
/****error******/
void fun(int a,int b)
{
  int t;
  /****error******/
  t=a;a=b;b=t;
}
main()
{
  int x,y;
  x=1;
  y=2;
  fun(&x,&y);
  printf("the result is %d,%d\n",x,y);
}
```

17.6.3　程序设计题

学生的记录由学号和成绩组成，M 名学生的数据已在主函数中放入结构体数组 stu 中，请编写函数 fun，它的功能是把小于等于平均分的学生数据放在 l 所指的数组中，大于等于平均分的学生人数通过形参 n 传回，平均分通过函数值返回。

请勿改动主函数 main 与其他函数中的任何内容，仅在函数 fun 的花括号中填入所编写的若干语句。

注意：部分源程序给出如下。

```
#include <stdio.h>
#define M 10
typedef struct
{
  char num [10];
  double s;
}SCORE;
double fun(SCORE *c,SCORE *l,int *n)
{

}
main ()
```

```
{
  SCORE stu[M]={{"03",76},{"02",69},{"04",85},{"01",91},{"07",72},{"08",64},{"06",87},{"09",
  60},{"11",79},{"12",73}};
  SCORE low [M],t;
  FILE *out;
  int i,j,n;
  double ave;
  ave=fun(stu,low,&n);
  printf("The %d student data which is lower than %7.3f:\n",n,ave);
  for(i=0;i<n;i++)
    printf("%s %4.1f\n",low[i].num,low[i].s);
  printf("\n");
  out=fopen("outfile.dat","w");
  fprintf(out,"%d\n%7.3f\n",n,ave);
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      if(low[i].s<low[j].s)
      {
        t=low[i];
        low[i]=low[j];
        low[j]=t;
      }
  for(i=0;i<n;i++)
    fprintf(out,"%4.1f\n",low[i].s);
  fclose(out);
}
double fun(int m)
{

}
main()
{
  int m;
  double f;
  FILE *out;
  printf("\nInput m:  ");
  scanf("%d",&m);
  f=fun(m);
  printf("\n\nf=%f\n\n",f);
  out=fopen("outfile.dat","w");
  for(m=0;m<10;m++)
    fprintf(out,"%f\n",fun(m+20));
  fclose(out);
}
```

17.7　上机模拟试题一分析与讲解

17.7.1　程序填空题

【答案】（1）*p==t　　（2）str[n++]=i　　（3）n

【解析】（1）本题中函数的功能是查找 s 中值为 a 的元素。

（2）从已给部分源程序的 main 主函数入手，“n=fun(s,ch);”语句调用 fun 函数，实现本题要求。

（3）进入 fun 函数，利用 while 循环对字符串中的每一个字符进行判断。

第 1 空：判断字符串 s 中的字符是否等于输入的字符 ch，根据 while 循环中的循环条件“*p”和 fun 函数中的 t，p 定义、赋值，得知这里的 if 判断表达式应该是“*p==t”。

第 2 空：如果符合上面的 if 判断表达式，这里应该根据题干要求将字符标识存储在数组 str 中，即“str[n++]=i”。

第 3 空：函数 fun 返回的是在字符串中查找到相同字符的个数，根据语句“str[n++]=i;”知道，相同的字符个数为数组 str 的大小，即第 3 空填“n”。

17.7.2　程序改错题

【答案】（1）switch(m)　　（2）case1:

【解析】本题中函数的功能是用递归算法计算斐波拉契级数数列中第 n 项的值。本题主要是对于 switch case 语法的考查。

（1）第一个标识下的 switch 后用括号括起来的表达式的标准语法后面是没有“;”的，所以“switch(m);”应该改为“switch(m)”。

（2）第二个标识下的 case 加常量表达式后面用冒号连接选择语句，所以“case1;”改为“case1:”。

17.7.3　程序设计题

【答案】

```
int i,t,j=0,*p=b;
t=p[0];           /*设置临时变量 t 初值指向第一个数组值*/
for(i=0;i<=m;i++)
  if(t==p[i])     /*当临时变量与数组中的某个值相同则跳出，比较下一个元素*/
   ;
  else            /*如果临时变量与数组中值不同，则对临时变量重新赋值*/
    {
      b[j]=t;
      t=p[i];
      j++;
    }
if(i>=m)
  b[j]=t;
return j;
```

【解析】该程序功能是删去一维数组中所有相同的数，使之只剩一个。解题思路是，首先在函数中定义临时变量指向每一个元素，然后在循环过程中将临时值和其他元素进行比较，如果相同，那么跳过相同字符。

（1）首先，设置一个临时变量，初值设置为一维数组的第一个元素。

（2）然后，依次将数组元素与临时变量比较，如果相同，则继续取数组下一个元素与临时变量比较；如果不同，则将临时变量的值赋值到数组的第一个元素，同时将与临时变量不同的数组元素赋值给临时变量。

（3）循环执行，直到所有的数据处理完毕。

17.8 上机模拟试题二分析与讲解

17.8.1 程序填空题

【答案】（1）temp2[j]=str[i]　　（2）str[i]=temp1[i]　　（3）j

【解析】本题中函数的功能是将指定字符串中的所有非数字字符移到所有数字字符之后。对于字符串中某种字符的操作，通常首先要执行的都是对字符串进行遍历，查找符合题目要求的字符。从已给部分源程序的 main 主函数开始入手，用“fun(str);”语句调用 fun 函数以实现本题要求。

第 1 空：首先函数对字符串进行遍历，判断字符是否为非数字字符，若是，则将该字符存入串 temp2 中，否则存入串 temp1 中，所以第 1 空填“temp2[j]=str[i]”。

第 2 空：根据题目要求，将数字字符放在串的前面，所以要先将存放数字字符的串 temp1 的内容首先赋值给字符串 str，所以这里“str[i]=temp1[i]”。

第 3 空：这个 for 循环将存放非数字的串 temp2 的内容，接在 temp1 赋值后，继续赋值给 str，从而形成非数字字符在数字字符的后面，根据生成 temp2 的循环，知道 temp2 中字符的个数为 j 个，所以这里填“j”。

17.8.2 程序改错题

【答案】break;

【解析】本题中函数的功能是计算并输出 max 以内最大的 10 个素数之和。其中，关于素数是指只能被 1 和本身整除的正整数（>1），所以判别 n 是否为素数，只要用 2，3，……，n-1 这些数逐个去除 n，观察余数是否为 0 即可，只要有一次相除余数为 0，n 就不是素数，否则 n 为素数。

在判断素数的循环过程中，只要被判断的数能被某数整除就不是素数，就会跳出判断循环，在 C 语言中，使用 break 语句跳出循环，并且 C 语言中的语句使用分号结尾。所以“break”应改为“break;”。

17.8.3 程序设计题

【答案】

```
int i,j,k=0;
for(i=0;i<M;i++)
  {
    for(j=0;j<N;j++)
      if(*(*(str+i)+j))  /*如果不指向行尾*/
        {
          a[k]=*(*(str+i)+j);  /*将行中的值赋值给数组 a*/
          k++; /*数组下标加 1*/
        }
      else /*如果指向尾，则跳出这一行*/
        break;
    a[k]='\0'; /*数组加尾符*/
  }
```

【解析】该程序功能是将放在字符串数组中的 M 个字符串（每串的长度不超过 N）按顺序合并组成一个新的字符串。其中，存储在二维字符数组中的每一行都是一个字符串；在每个字符串的结尾处，各有一个字符串结束符‘\0’。

（1）二维字符数组第 i 行第 j 列元素 str[i][j]，就是指针形式的*(*(str+i)+j)。因此，如果字符*(*(str+i)+j)是串的结束符，则说明该字符串已经结束。

（2）当一维字符数组存放完字符串中各字符后，必须在一维字符数组的尾字符处加一个结束符‘\0’，这样才能把字符数组变成字符串。否则，字符数组就只能是非字符串的一般字符数组。

17.9　上机模拟试题三分析与讲解

17.9.1　程序填空题

【答案】（1）j=2　　（2）i　　（3）j

【解析】本题中函数的功能是求所有小于等于 x（x>2）的素数的个数。

此类题的解题思路是：首先判定素数，然后根据判断结果进行统计。从已给部分源程序的 main 主函数开始入手，用“result=fun(x);”语句调用 fun 函数，实现本题要求。

第 1 空：根据素数的解题思路，素数从 2 开始，所以这里应该填“j=2”。

第 2 空：for(i=2;i<=x;i++)循环中判断 i 是否为素数，所以第 2 空填“i”。

第 3 空：是对 i 为素数的个数的统计，所以这里填“j”。

17.9.2　程序改错题

【答案】（1）if(m==0)　　（2）result*=m--;

【解析】本题中函数的功能是计算 m 的阶乘。从已给定源程序的 main 主函数开始入手，从键盘输入 m，然后通过“printf("\n\n%d!=%1f\n\n",m,fun(m));”语句调用 fun 函数。

（1）第 1 个标识下的“if m==0”是判断输入 m 值为 0 时执行的操作，在 C 语言中 if 判断表达式应该用括号括起来，所以应将“if m==0”改为“if(m==0)”。

（2）第 2 个标识下“result=m--;”应该求得 m 的阶乘，而这里的语句只是将 m 的值赋给 result 并执行减 1 操作，所以应该填“result *=m--;”，执行 result 和 m 的乘法，然后 m 执行减 1 操作。

17.9.3　程序设计题

【答案】

```
int i,j=0,n=0,max;
max=p[0].s;
for(i=0;i<M;i++)
  if(p[i].s>max)
     max=p[i].s;
for(i=0;i<M;i++)
  if(p[i].s==max)
  {
     *(q+j)=p[i];
     j++;
     n++;
  }
return n;
```

【解析】该程序功能是把分数最高的学生数据放在 high 所指的数组中。本题是关于求解结构体中某些成员的最大值的，首先将第一个值设定为最大值，并在循环中将其他所有值与该值进行比较，求得最大值。然后将最大值与所有值进行比较，求得所有的最大值。

（1）要把最高分数的学生数据放在数组中，首先求出最高分数，使用一个 for 循环使所有的成绩进行比较，找出最高的分数来。

（2）找出所有与最高分数相等的学生，并将最高分数的人数累加。

17.10 上机模拟试题四分析与讲解

17.10.1 程序填空题

【答案】（1）i=1　　（2）&&　　（3）--j

【解析】本题中函数的功能是求能整除 k 的偶数。此类题的解题思路是在循环中判断能整除 k 的数的基础上增加对其是偶数的判断。从已给部分源程序的 main 主函数开始入手，从键盘输入整数 k，用“fun(k,a);”语句调用 fun 函数，实现本题要求。

第 1 空：求能整除 k 的偶数，应该从 1 开始到 k，逐个对其判断，所以这里循环条件中应补全“i=1”。

第 2 空：根据题干要求，当前的 i 不但要能整除 k，而且要求它是偶数，所以这里的逻辑关系应该是“&&”。

第 3 空：在 fun 函数中，通过 for 循环输出结果，根据已给出的循环条件中的已有代码“i>=0;i--”，可知，这个循环是从最后一个元素到第一个元素的输出，根据结果是数组 a 中元素的个数，所以第 3 空填“--j”。

17.10.2 程序改错题

【答案】void top(char *s)

【解析】本题中函数的功能是将其中每个单词的第一个字母改成大写。根据题干中给出的提示：单词是指由空格隔开的字符串。确定单词中的第一个字母，也就是空格字符后面的第一个字符，然后将其改写为大写字母即可。

主函数中“top(str);”语句中变量 str 是数组的名称，所以 top 函数中的变量 s 应是指针型变量，应将“void top(char s)”改为“void top(char *s)”。

17.10.3 程序设计题

【答案】

```
int i=0;
char *q=s;
while(q<r)
  {
    s[i]=*q;
    q++;
    i++;
  }
while(q<v)
  {
    if(*q!='#')
    {
      s[i]=*q;
      i++;
    }
    q++;
  }
while(*q)
  {
    s[i]=*q;
    i++;
    q++;
  }
s[i]='\0';
```

【解析】该程序功能是除了字符串前导和尾部的#号之外，将串中其他#号全部删除。本题的解题过程首先对小于主

函数给出第一个字母的#号进行复制赋值，然后对小于最后一个字母内的所有字符进行判断，如果是#号则删除，否则复制赋值，最后对最后一个字母到串尾的#号进行复制赋值。

（1）设置一个指针变量 q 指向字符串的头位置，并使用循环，将从字符串头开始到第一个字母的#号复制到字符串 s。

（2）对第一个字母到最后一个字母之间的字符进行循环判断是否为#号，如果是则跳过，否则复制到字符串 s。

（3）将最后一个字母到串尾的#号复制到字符串 s，然后在新生成的字符串尾加‘\0’。

17.11　上机模拟试题五分析与讲解

17.11.1　程序填空题

【答案】（1）a[i][j]=0　　（2）a[i][j]=1

【解析】本题中函数的功能是将矩阵周边元素赋值 0，非周边元素赋值 1。其中，N×N 矩阵周边元素下标的特点是行、列为 0 及为 n-1。

第 1 空：“if(i==0||i==n-1||j==0||j==n-1)”语句对周边元素进行赋值，所以根据题干要求，第 1 空格填“a[i][j]=0”。

除去周边元素后，其他的就是非周边元素，对非周边元素赋值为 1，第 2 空填“a[i][j]=1”。

17.11.2　程序改错题

【答案】（1）void fun(int *a,int *b)　　（2）t=*a;*a=*b;*b=t;

【解析】本题中函数的功能是交换主函数中两个变量的值。利用临时变量存放临时交换值，实现两个变量的交换。

（1）主函数中“fun(&x,&y);”引用的是变量 x 和 y 的地址，所以第一个标识下的 fun 函数定义中的参数应该是指针型的，即“void fun(int a,int b)”改为“void fun(int *a,int *b)”。

（2）同理，第二个标识符下的“t=a; a=b; b=t;”改为“t=*a; *a=*b; *b=t;”。

17.11.3　程序设计题

【答案】

```
double aver=0.0;  /*定义平均分的变量设置初始化*/
int i,j=0;
for(i=0;i<M;i++)  /*求总和*/
  aver+=c[i].s;
aver/=M;  /*求平均分*/
for(i=0;i<M;i++)  /*循环判断每一个成绩与平均分的关系*/
  if(c[i].s<=aver)     /*如果成绩小于等于平均分*/
    {
      *(l+j)=c[i];     /*将小于等于平均分的成绩存入 l*/
      j++;
    }
*n=j;  /*将小于平均分的个数 j 赋值给 n*/
return aver;  /*返回平均分*/
```

【解析】该程序功能是把低于或等于平均分的学生数据放在 l 所指的数组中，高于等于平均分的学生人数通过形参 n 传回，平均分通过函数值返回。解题过程首先求得平均分，然后将所有成绩与平均分进行比较，如果小于或等于平均分，存入指定数组。

（1）通过循环求总分，然后求得平均分。

（2）在循环中，使平均分与每个成绩进行比较，并将满足条件的数据存入数组或对其个数进行累加。

附录 A　习题分析与解答

第 1 章　习题分析与解答

一、选择题

1. 答案：B

解析：数据的存储结构分顺序存储结构和链式存储结构，一个数据的逻辑结构可以有多种存储结构。顺序结构中数据元素所占的存储空间是连续的，而链式存储结构中，数据元素通过指针就联系在一起了，所以所占的存储空间不一定是连续的。

2. 答案：D

解析：队列（Queue）是指允许在一端进行插入，而在另一端进行删除的线性表。允许插入的一端称为队尾，允许删除的一端称为队头。在队列这种数据结构中，最先插入的元素将能够最先被删除；反之，最后插入的元素最后才能被删除。因此，队列又称“先进先出”或“后进后出”的线性表。

3. 答案：D

解析：各种排序方法中最坏情况需要比较的次数分别为：冒泡排序 N(N-1)/2、快速排序 N(N-1)/2、直接插入排序 N(N-1)/2、堆排序 $O(N\log_2N)$。

4. 答案：B

解析：栈是线性表的一种，它的特点是先进后出，并且是只能在表的一端进行插入和删除操作的线性表，入栈和出栈都是在栈顶进行的，因此具有记忆作用，栈可以采用顺序存储，也可以采用链式存储。

5. 答案：A

解析：在任意一棵二叉树中，度为 0 的结点（即叶子结点）总是比度为 2 的结点多一个，所以该二叉树的叶子结点数等于 n+1。

6. 答案：C

解析：二叉树前序遍历的简单描述为：若二叉树为空，则返回结点；否则（1）访问根结点，（2）前序遍历左子树，（3）前序遍历右子树。可见，前序遍历二叉树的过程是一个递归的过程。根据题目中给出的二叉树的结构可知前序遍历的结果是 ABDYECFXZ。

7. 答案：C

解析：对长度为 n 的线性表进行顺序查找时，从表中的第一个元素开始，给定的值与表中元素的关键字逐个进行比较，直到两者相符，查找完成。在最坏的情况下，要查找的元素是表的最后一个元素或无该元素，这两种情况都需要将这个表中的所有元素进行比较，因此比较次数为 n。

8. 答案：D

解析：数据的逻辑结构是指反映数据元素之间逻辑关系的数据结构。数据的逻辑结构在计算机存储空间中的存放形式称为数据的存储结构（也称数据的物理结构）。一般来说，一种数据的逻辑结构根据需要可以表示成多种存储结构，常用的存储结构有顺序、链接和索引等。

9. 答案：D

解析：算法分析是指对一个算法的运行时间和占用空间做定量的分析，计算相应的数量级。分析算法的目的就是要降低算法的时间复杂度和空间复杂度，提高算法的执行效率。

10. 答案：A

解析：队列是一种特殊的线性表示，只允许在表的一端插入元素，在表的另一端删除元素，插入元素的一端叫“队尾”，删除元素的一端叫“队头”，先插入的元素先被删除，是按先进先出的原则组织数据的。

11. 答案：C

解析：栈和队列都是操作受限的线性表，只允许在端点插入和删除，不同点是栈只允许在表的一端进行插入和删除操作，而队列允许在表的一端进行插入操作，而在另一端进行删除操作。

12. 答案：B

解析：在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据之间的逻辑关系是由指针域来

确定的。

13. 答案：A

解析：二分法查找只用于顺序存储的有序线性表，而顺序查找用在顺序存储的非有序线性表和线性链表。

14. 答案：A

解析：对线性表进行检索，要求线性表是按顺序方式存储的，并按关键码的大小排好序，而不按关键码检索频率排序。

15. 答案：B

解析：根据二分法查找的查找过程，首先将 90 与表中的中间元素 50 进行比较，由于 90 大于 50，所以在线性表的后半部分查找。第二次比较的元素是后半部分的中间元素，即 90，这时两者相等，即查找成功。

16. 答案：A

解析：线性表的存储通常有两种存储结构：顺序存储结构和链式存储结构。

17. 答案：B

解析：快速排序的基本思想是：经过一趟排序待排序记录分割成独立的部分，其中前半部分元素都比基准元素小，而后半部分元素都比基准元素大，再分别对这两个部分的记录继续进行排序，以达到整个序列有序。在上述序列中，比关键码 33 小的元素有 4 个，因此第一趟排序完成后关键码 33 被放到第 5 个数的位置上。

18. 答案：C

解析：数据的存储结构有顺序存储结构和链式存储结构两种，不同存储结构的数据处理效率不同。由于链表采用链式存储结构，元素的物理顺序并不连续，对于插入和删除无须移动元素，很方便，但当查找元素时就需要逐个元素查找，因此查找的时间相对更长。

19. 答案：B

解析：当数据表 A 中每个元素距其最终位置不远，说明数据表 A 按关键字值的排序是有序的。在待排序列基本有序的情况下，采用插入排序所用的时间最少。

20.答案：B

解析：链表采用的是链式存储结构，它的结点空间可以动态申请和释放；它的数据元素的逻辑次序靠结点的指针来指示，插入删除不需要移动数据元素。但是链式存储结构也有不足之处，每个结点中的指针域需额外占用存储空间，它是一种非随机存储结构。

二、填空题

1. 答案：有穷性

解析：算法的 3 个基本特征包括有穷性、确定性和可行性，其中，算法的有穷性指的是算法必须在有限的时间内完成，即算法必须在执行有限个步骤之后终止。

2. 答案：4

解析：当对表进行二分法查找时，先用 12 和表中间的元素 15 进行比较，接着与数据元素 8 进行比较，再与数据元素 10 进行比较，最后与 12 进行比较，查找成功，所以需要的比较次数为 4。

3. 答案：栈顶

解析：栈是限定在表的一端进行插入和删除操作的线性表。在表中，允许插入和删除的一端叫做“栈顶”，不允许插入和删除的一端叫做“栈底”。

4. 答案：顺序存储（顺序方式存储）

解析：二分法查找对表的要求是有序顺序表，即第一要求是数据元素有序，第二要求是按顺序方式存储。

第 2 章　习题分析与解答

一、选择题

1. 答案：C

解析：本题考查对面向对象的理解，面向对象的程序设计是对象模拟问题领域中的实体，各对象之间相互独立，相互依赖性小，通过消息来实现对象之间的相互联系。

2. 答案：C

解析：对象的封装性是指从外部看只能看到对象的外部特征，即只需知道数据的取值范围和可以对该数据施加的操作，而不需要知道数据的具体结构及实现操作的算法。对象的内部，即处理能力的实行和内部状态对外是不可见的，从外面不能直接使用对象的处理能力，也不能直接修改其内部状态，对象的内部状态只能由其自身改变。

3. 答案：B

解析：源程序的文档化主要包括 3 点：①符号名应具有一定含义，便于理解程序功能；②正确的程序注释；③良好的视觉组织：利用空格、空行、缩进等技巧使程序层次清晰。

4. 答案：D

解析：类（chass）描述的是具有相似属性与操作的一组对象，类是具体对象的实例。

5. 答案：D

解析：在程序设计中，程序流程图是必需的，程序的正确注释有助于读者理解程序，不是可有可无的。程序的结构化与模块化都是程序设计的基本原则，它们不存在矛盾的地方，程序要求模块化，每个模块要符合结构化原则。

6. 答案：B

解析：程序的 3 种基本控制结构包括顺序、选择和重复（循环），这 3 种结构就足以表达出各种其他形式的结构。

7. 答案：D

解析：在面向对象的方法中，对象之间能通过消息进行通信，消息中包含传递者的要求，它告诉接收者需要做

哪些处理，但并不指示接收者应该怎么完成这些处理，接收者独立决定采用什么方式完成所需的处理。

8. 答案：C

解析：面向对象的设计方法的基本原理是：使用现实世界的概念抽象地思考问题从而自然地解决问题。它强调模拟现实世界中的概念而不强调算法，它鼓励开发者在软件开发的绝大部分中都应用领域的概念去思考。

9. 答案：D

解析：面向对象设计方法与面向过程设计方法有本质不同，其基本原理是，使用现实世界的概念抽象地思考问题从而自然地解决问题，其特点包括继承性、封装性、多态性等。模块化是结构化程序设计的特点。

二、填空题

1. 答案：面向对象

解析：面向对象的程度设计方法中涉及的对象是系统中用来描述客观实体的，是构成系统的一个基本单位。

2. 答案：封装性

解析：对象具有以下 3 个基本特点：继承性、封装性、多态性。其中，封装性是指从外面看只能看到对象的外部特征，对象的内部特征即处理能力的实行和内部状态，对外是不可见的，对象的内部状态只能由其自身改变。

3. 答案：对象

解析：将操作相似的对象归为类，也就是说，类是具有共同属性、共同方法的对象的集合。

4. 答案：程序设计风格

解析：程序设计风格是指编写程序时所表现出的特点、习惯和逻辑思路。

第 3 章　习题分析与解答

一、选择题

1. 答案：D

解析：需求分析常用的工具有 4 种：数据流图（DFD）、数据字典（DD）、判断树和判定表。PAD（问题分析图）、PFD（程序流程图）、N-S（盒式图）都是详细设计的常用工具，不是需求分析的工具。

2. 答案：A

解析：软件设计的步骤分为两部分，一部分是概要设计（总体设计），另一部分是详细设计。

3. 答案：D

解析：本题考查对软件生命周期的理解，软件生命周期整体上可分为定义阶段、开发阶段和维护阶段。其中定义阶段包括可行性研究、计划制订和需求分析；开发阶段包括概要设计、详细设计、开发和测试；维护阶段是一个单独阶段，不包含在开发阶段内，它是所花费用最多的一个阶段。

4. 答案：B

解析：需求分析的最终结果是生成软件需要的规格说明书，可以为用户、分析人员和设计人员之间的交流提供方便，可以直接支持目标软件系统的确认，又可以作为控制软件开发进程的依据。

5. 答案：C

解析：软件工程是建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。软件工程的主要思想是强调在软件开发过程中需要应用工程化原则。

6. 答案：D

解析：DFD（数据流图，Data-Flow-Diagram）是描述数据处理过程的工具，是需求理解的逻辑模型的图形表示，它直接支持系统功能建模。在软件详细设计的过程阶段，要对每个模块规定的功能及算法的设计给出适当的算法描述。常见的过程设计工具有：1 图形工具，即程序流程图、N-S、PAD、HTPO；2 表格工具，即判定表；3 语言工具，即 PDL（伪码）。

7. 答案：D

解析：程序调试是由程序开发者完成诊断和改正程序中的错误的过程，软件测试是由专门的测试人员完成，是为发现错误而执行程序的过程。软件维护是指软件系统交付使用以后，为了改正错误或满足新的需要而修改软件的过程，是软件生存周期中非常重要的一个阶段。

8. 答案：D

解析：软件工程包括 3 个要素，即方法、工具和过程。方法是完成软件工程项目的技术手段；工具支持软件的开发、管理和文档生成；过程支持软件开发的各个环节的控制、管理。

9. 答案：C

解析：软件工程管理主要包括软件管理学、软件工程学、软件心理学等内容。软件经济学是研究软件开发中成本的估算、成本效益分析的方法和技术，用经济学的基本原理来研究软件工程开发中的经济效益问题。

10. 答案：B

解析：需求分析是软件定义时期的最后一个阶段，可以概括为 4 个方面：①需求获取，②需求分析，③编写需求规格说明书，④需求评审。

11. 答案：B

解析：本题是对设计软件结构的考查。设计软件结构是在软件概要设计阶段进行的，而概要设计属于软件开发期。

12. 答案：A

解析：数据流图的主要图形元素有 4 种：①加工，用圆或椭圆表示，输入数据经过加工变换产生输出；②数据流，用箭头表示，是沿箭头方向传送数据的通道；③存储

文件，用双杠表示，是处理过程中存放各种数据的文件；④外部实体用方框表示，是系统和环境的接口。

13. 答案：C

解析：程序设计不仅仅是编制程序，还包括文档开发、程序测试、程序调试等工作。一般来说程序测试不由程序员自己完成。程序修改过程可能带来新的错误，调试改错后需要进行回归测试，以确认错误是否排除，是否引入新的错误。

14. 答案：D

解析：在程序设计中，各模块间的内聚性越强，则耦合性越弱。一般较优秀的软件设计，应尽量做到高内聚、低耦合，有利于提高模块的独立性。耦合性与内聚性是模块独立的两个定性标准，是互相关联的。在 4 个选项中数据耦合独立性最强。

15. 答案：D

解析：需求分析是软件定义时期的最后一个阶段，它的基本任务就是详细调查现实世界要处理的对象，充分了解原系统的工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。

16. 答案：C

解析：软件定义、软件开发、软件运行维护组成了软件的生命周期。其中软件定义阶段的主要工作是可行性研究、计划制订和需求分析等；软件开发阶段的主要工作有概要设计、详细设计和测试等；运行维护阶段的主要工作是软件的运行及后期的维护等。

二、填空题

1. 答案：软件生命周期

解析：软件产品从提出、实现、使用维护到停止使用退役的过程称为软件生命周期，可以将软件生命周期分为软件定义、软件开发及运行维护 3 个阶段。

2. 答案：黑盒

解析：等价类划分法是一种典型的黑盒测试方法。它将程序的所有可能输入数据分成若干部分，然后从每个等价类中选取数据作为测试用例。

3. 答案：软件开发

解析：软件生命周期分为软件定义期、软件开发期和软件维护期。软件定义包括问题定义、可行性研究和需求分析 3 个阶段；软件开发期包括系统设计、详细设计、编码和测试 4 个阶段；软件维护即运行和维护阶段。

4. 答案：相关文档

解析：所谓计算机软件是计算机系统中与硬件相互依存的另一部分，是包括程序、数据及相关文档的完整集合。

5. 答案：软件工程过程

解析：本题考查对软件工程的理解，软件工程环境被称为软件工程过程（或软件开发环境），全面支持软件开发过程的软件工具集合。

6. 答案：降低

解析：模块化是指解决一个复杂问题时，自顶向下逐层把软件系统划分成若干模块的过程，由此分解来降低程序设计的复杂性。

第 4 章　习题分析与解答

一、选择题

1. 答案：C

解析：数据定义语言 Data Definition Language（DDL）是 SQL 语言的一部分，用来定义概念模式、内模式和外模式。

2. 答案：D

解析：数据库设计目前一般采用生命周期法，即把整个数据库应用系统的开发分解成目标独立的 4 个阶段。它们分别是需求分析阶段、概念设计阶段、逻辑设计阶段和物理设计阶段。

3. 答案：A

解析：在数据库系统中，物理独立性是指数据的物理结构（包括存储结构、存取方式等）的改变，如存储设备的更换、物理存储的更换、存取方式的改变等都不影响数据库的逻辑结构，从而不致于引起应用程序的变化。

4. 答案：B

解析：数据库中的数据具有集成、共享的特点，即数据库集中了各种应用的数据，进行统一的构造与存储，而使它们能被不同的应用程序所使用，因而数据库技术的根本目标是解决数据的共享问题。

5. 答案：C

解析：关键字是指属性的组合，其值能够唯一地标识一个元组，而在 SC 中学号和课号的组合可以对元组进行唯一的标识。

6. 答案：C

解析：数据库系统（DBS）由数据库（DB）、数据库管理系统（DBMS）、数据库管理员、硬件平台和软件平台 5 个部分组成，可见 DB 和 DBMS 都是 DBS 的组成部分。

7. 答案：D

解析：在关系中凡能唯一标识元组中最小属性集的称为该关系的单键或码。二维表中可能有若干个键，它们称为该表的候选码或候选键。从二维表的所有候选键中选取一个作为用户使用的键称为主键或主码。

8. 答案：B

解析：投影运算是指，选取关系中的某些列（属性），并将得到的结果中的重复元组消去。

9. 答案：B

解析：将 E-R 图转换成指定 DBMS 中的关系模式是数据库逻辑设计的主要工作。从 E-R 图到关系模式，实体

和联系都可以表示成关系。

10. 答案：A

解析：在数据库设计过程中，独立于计算机有硬件与 DBMS 软件的设计阶段是概念设计阶段。

11. 答案：B

解析：数据库的物理设计的主要目的是对数据库内部物理结构做调整并选择合理的存取路径，以提高数据库的访问速度及有效利用的存储空间。

二、填空题

1. 答案：数据库系统

解析：在人工管理阶段，数据无法共享，冗余度大，完全依赖于程序；在文件系统阶段，数据共享性差；而数据库系统是具有三级模式及二级映射的抽象结构系统，从而保证了数据独立性的实现。

2. 答案：物理独立性

解析：数据的独立性是指数据和应用程序相互独立，分为物理独立性和逻辑独立性。其中物理独立性是指数据的物理结构改变时，不需要修改应用程序，而逻辑独立性是指当逻辑结构改变时，不需要修改应用程序。

3. 答案：数据库管理系统

解析：数据库管理系统（DBMS）是数据库的管理机构，负责数据库中的数据组织，数据操纵，数据维护、控制及保护，以及数据服务等。

4. 答案：投影

解析：关系数据库的专门关系运算包括选择、投影和连接 3 种。其中投影是从二维表的列方向上进行的运算，而选择和连接是从二维表的行方向上进行的运算。

5. 答案：逻辑数据模型

解析：数据模型按不同层次分成概念数据模型、逻辑数据模型和物理数据模型 3 类。

第 5 章　习题分析与解答

一、选择题

1. 答案：C

解析：每个 C 程序有且只有一个主函数 main()，且程序必须从 main()函数开始执行，并在 main()函数中结束。

2. 答案：A

解析：在 C 语言中，C 程序可以由多个程序文件组成，也可以由一个或多个函数组成，一个 C 函数可以单独作为一个 C 程序文件存在，它可以实现多种算法。

3. 答案：D

解析：源程序经过 C 编译程序编译之后生成一个后缀为.obj 的二进制文件（称为目标文件），然后由称为“连接程序”的软件，生成一个后缀为.exe 的可执行文件。

4. 答案：B

解析：由 C 语言构成的指令序列称为 C 源程序，源程序文件的后缀名为.c。源程序经过 C 编译程序编译生成后缀为.obj 的二进制文件（称为目标文件），然后由称为“连接程序”的软件，把目标文件与 C 语言提供的各种库函数连接起来，生成后缀为.exe 的可执行文件。

5. 答案：C

解析：本题涉及 C 语言的 3 个概念：（1）C 语言是区分大小写的，例如，q 和 Q 是两个不同的变量；（2）变量的实质就是在内存中占据一定的存储单元，存储单元里存放的是该变量的值，变量的值可以根据需要进行修改；（3）整数在允许的范围内可以准确地表示出来，但不可能表示无限精确的实数。正整数可用二进制、八进制、十进制和十六进制表示。

6. 答案：D

解析：每个 C 程序有且只有一个主函数（main），且程序必须从 main()函数开始执行，但是 main()函数可以放在程序中的任意位置。

7. 答案：D

解析：每个 C 程序有且只有一个主函数 main()，且程序必须从 main()函数开始执行，但是 main()函数可以放在程序中的任意位置。

8. 答案：A

解析：C 语言是函数式的语言，它的基本组成单位是函数，在 C 语言中任何程序都是由一个或者多个函数组成的。

9. 答案：A

解析：本题涉及 C 语言最基本的两个知识点：（1）C 程序是由函数构成的，有且仅有一个主函数，也可以有其他的函数；（2）整数在允许的范围内可以准确无误地表示出来，但计算机的存储能力有限，不能表示无限精确的实数。

10. 答案：C

解析：本题涉及 C 语言的 4 个概念：（1）主函数 main()可以放在程序中的任意位置；（2）C 语言的书写格式自由，一行可写多条语句，一条语句也可写在不同行上；（3）main()后面括号是填写函数的参数，可以是空，但括号必须得有；（4）程序的注释仅仅供阅读之用，并不参与程序的编译，所以编译也不会发现注释中的错误。

二、填空题

1. 答案：顺序结构

解析：结构化程序有 3 种基本结构，即顺序结构、选择结构（包括 if 语句和 switch 语句）和循环结构（包括 for 语句、while 语句、do while 语句）。

2. 答案：一条语句　分号（或;）

解析：按 C 语法规定，在程序中，用一对花括号把若干语句括起来的称为复合语句，复合语句在语法上被认为

是一条语句。空语句的形式是：一个分号（;），它由一个单独的分号构成，程序遇到空语句时，不产生任何动作。

3. 答案：分号“;”

解析：按 C 语法规定，C 语言中所有语句都必须由一个分号（;）结束。

第 6 章　习题分析与解答

一、选择题

1. 答案：A

解析：赋值运算符=的右边既可以是常量、变量，也可以是函数调用或表达式，但左边必须是变量，不能为常量和表达式，因此选项 A 是正确的，选项 C、D 是错误的。选项 B 中，运算符“%”之后只能是整数，所以错误。

2. 答案：A

解析：C 语言中规定标识符只能由字母（大小写均可，但区分大小写）、数字和下画线 3 种字符组成，并且第一个字符必须为字母或者下画线。选项 B、D 中出现非法字符，D 中空格不能出现在一个标识符的中间，故选 A。

3. 答案：C

解析：在表达式中，其+=，-=的优先级相同，按从右到左的结合方向运算，而*的优先级最高，表达式写成 a=a+(a-(a*a))代入 3 可得-12。

4. 答案：D

解析：因为 x 的值为大于 1 的奇数，所以 x 除以 2 的余数等于 1，因此，选项 A、C 中表达式的结果为真，不为 0；对于选项 B 来说，x 除以 2 的商不会等于 0；选项 D 中表达式的结果为假，即等于 0。

5. 答案：B

解析：C 语言规定，标识符是由字母、数字或下画线组成，并且它的第一个字符必须是字母或者下画线，不能使用关键字，A 中 void，C 中 if，D 中 do 都属于关键字。

6. 答案：A

解析：“int k;”这条语句用于定义一个整型变量 k，并对其进行初始化赋值操作。

7. 答案：C

解析：对同一数据类型变量的定义，不同变量之间用“,”分隔，变量与变量类型说明中有一空格。因此，选项 A 是错误的，b 前面不该用分号；选项 D 是错误的，变量类型说明后面不应用逗号。变量定义时赋值不能用连等形式，因此选项 B 错误。

8. 答案：C

解析：赋值运算符的左边必须是一个代表某一存储单元的变量名，而 A 选项中的“y*5=x+z”部分是非法赋值。求余运算的对象只能是整型，故选项 B 和选项 D 是错误的。选项 C 为逗号表达式。

二、填空题

1. 答案：3

解析：本题中的表达式为逗号表达式，其运算顺序为从左到右，最后一个表达式的值为该逗号表达式的值。先执行 n=i=2，得 i=2；然后执行++i，得 i=3；最后执行 i++，此时表达式先取 i 的值为 3，然后进行加 1 运算，i 的值为 4。

2. 答案：67 G

解析：根据题意，a='A'+'5' '-'3'=65+2=67，b=67+4=71，71 即为字母 G 的 ASCII 码值。要求程序按十进制数形式输出 a，按字符形式输出 b，所以结果为 67 G。

3. 答案：a=14

解析：本题考查的是表达式的优先级问题。先计算表达式 3*5=15，再计算 a+4=14,将数据 14 赋值给 a，根据 printf()函数内的输出格式控制串，最后的输出结果应为“a=14”。

4. 答案：11　11

解析：赋值表达式 m=012 中的常数 012 为八进制数，变成十进制数后为 10，++m 后以十进制数格式%d 输出后为 11。

5. 答案：32

解析：本题考查自加运算符++和自减运算符--。++和--的几个特点：（1）只能作用于变量，不能用于表达式或常量；（2）前缀形式是在使用变量之前先将其值加 1 或减 1，后缀形式是先使用变量原来的值，使用完后再使其加 1 或减 1。题中：++c 计算时 c=4，b++计算时 b=2，18+(b++)-(++c)=16，a*=16，得 a=32。

第 7 章　习题分析与解答

一、选择题

1. 答案：B

解析：本题考查 scanf 函数，输入格式符%2d%f 表示只接收了字符，其中 a 输入的值为 2 位的十进制整数，即 87，然后将紧接着的 6 赋给 b，因为 b 的值为单精度数，即 6.000000。

2. 答案：D

解析：在 C 语言程序中，可以用一个符号名来代表一个常量，称为符号常量。这个符号名必须在程序中进行特别的“指定”，并符合标识符的命令规则。在本题中圆周率 π 是一个符号常量，但在程序中并没有指定其值，所以编译器找不到其值就会报错。因此，选项 D 正确。

3. 答案：D

解析：“double *p ,a;”语句表示定义了一个指向双精度型的指针变量 p 和双精度型变量 a，而语句“p=&a”表示将

变量 a 的地址赋给指针变量 p；scanf("%2d%f ",p)表示用键盘输入的数赋给指针变量 p 指向的地址单元。scanf()函数要求输入 double 型数据时，格式控制符必须用%1f（或%1e），否则数据不能正确输入。所以选项 D 为正确答案。

4. 答案：A

解析：C 语言规定，在字符“/*”中间的部分是注释内容，且注释部分的内容不参与程序的编译和运行，因此，本题中程序语句“b=2;”没有执行，故本题的答案为选项 A。

5. 答案：B

解析：%u 格式符，表示以十进制形式输出无符号整型变量。本题中无符号整型变量 x=0xFFFF（十六进制）表示的是无符号整型变量（十进制形式）的最大值 65535，所以最后输出的结果为 65535。

6. 答案：D

解析：格式字符 X 或 x 是以十六进制无符号形式输出整型数（注：输出时不显示前导 0x 或 0X）。

7. 答案：A

解析：C 语言规定，赋值符号的右边可以是一个赋值表达式，因此 C、D 正确；在选项 B 中，a++是一个自加 1 的表达式，a 被重新赋值，因此它是一个合法的赋值表达式；选项 A 中，a+d 是一个算术表达式，虽然最后有一个分号，但这个表达式中没有赋值操作，因此它不是一条赋值语句。

8. 答案：D

解析：选项 A 是一个合法的赋值表达式，但因为结尾没有加分号，所以它不是一个赋值语句；选项 B 赋值号右边的强制类型转换是错误的，正确的形式是“(int)(a+b)”；选项 C 是一个逗号表达式，但结尾也没有分号，因此也不是语句；选项 D 是一个自加运算符构成的赋值表达式，且末尾有分号，所以它是合法的赋值语句。

9. 答案：D

解析：选项 A 是一个合法的赋值表达式，但结尾没加分号，所以它不是一个赋值语句；选项 B 是一个逗号表达式，也因为结尾没有分号而不是合法的赋值语句；选项 C 是一个算术表达式，虽然有分号，但这个表达式没有赋值操作，因此不是一条赋值语句。

10. 答案：D

解析：在 C 语言中语句结束时应该以分号结尾，程序中的第 1 个“printf("###")”输出语句后面少一个分号“;”，因此执行程序时会出错。

11. 答案：A

解析：选项 A 是一个表达式，它后面没有分号结尾（C 语言中规定，语句必须以分号结束），所以它不是语句；选项 B 用一个花括号把几条语句括号起来了，这是一个复合语句；选项 C 中只有一个分号，是一个空语句；选项 D 是一个复合语句，也是一个空语句。

12. 答案：C

解析：本题考查语句的基本构成。选项 A 中 j=5 后面少了一个分号；选项 B 中少了“}”；选项 D 不是一个完整的函数定义格式，一个完整的函数定义格式还包括一对花括号；选项 C 正确，是一个空语句。

13. 答案：A

知识拓展：printf 函数的调用形式如下：“printf（格式字符串，输出项表）;”，其功能是：按格式字符串中的格式一次输出输出项表中的各输出项。%f 表示以 float 类型输出数据。

解析：本题看起来好像挺复杂的，其实就是对表达式(int)(x*1000+0.5)/(float)1000 进行运算的过程。运算过程分为几下几步：第 1 步，因为括号()的优先级高于强制类型转换，所以先计算表达式(x*1000+0.5)的值再进行类型转换，括号中先计算 x*1000 值为 1236.547，然后将 0.5 也转换成 float 型，再进行相加得 1237.047；第 2 步，强制转换为 int 型，得到 1237；第 3 步，将 1237 再转换成 float 型和(float)1000 相除，得到 float 型值 1.237000。

14. 答案：C

解析：本题考查 printf 函数的常用输出格式。“%.2e”中的“e”表示以指数形式输出数据，“%.2”表示输出结果要输出两位小数，后面的数值进行四舍五入。

15. 答案：D

解析：C 语言规定，输出 long 型数值的格式符为%1d。

16. 答案：C

解析：C 语言中，%d 表示输出带符号的十进制整数，%x 表示以十六进制无符号型输出整型数据（即不带前导 0x 或 0X），%o 表示以八进制无符号型输出整型数据（即不带前导 0）

17. 答案：A

解析：本题考查 printf 函数的格式。printf 函数中格式说明符之前插入的任何字符都原样输出。格式说明与输出项的个数也要相等，如果格式说明的个数少于输出项的个数，则对于多余的输出项不予输出。本题格式说明的个数是 1，输出项的个数是 1，所以对于插入的 5222 原样输出，对于输出项 a 也输出。

18. 答案：D

解析：本题考查 printf 函数的格式。“%8x”表示以十六进制无符号形式输出整型数据。“8”表示指定输出数据的宽度为 8 位。

19. 答案：D

解析：本题考查 scanf 函数的基本格式。当需要从键盘上输入数据时，输入的数值之间需要有间隔符（空格符号、制表符号、回车符号）。间隔符号的使用数量不限，直到按

下＜回车＞键，scanf 函数才会接受从键盘输入的数据。

20. 答案：D

解析：scanf 函数的调用形式是：scanf（格式字符串，输入项地址表）。其中，“格式字符串”是要输入的变量的格式符；“输入项地址表”是要输入的变量的地址。题中定义变量 a 为双精度型变量，双精度变量的格式符为“1e”；变量的地址用取地址符“&”加变量名表示，例如变量 a 的地址为“&a”。

21. 答案：B

解析：scanf 函数的调用形式是 scanf（格式字符串，输入项地址表）。其中，“格式字符串”是要输入的变量的格式符，则在输入时要求按一一对应的位置原样输入这些字符，其中的逗号也必须输入。

22. 答案：B

解析：在 scanf 函数的格式控制中，格式说明的类型与输入项的类型必须一一对应匹配，如不匹配将导致数据输入出现错误，但是系统并不报错。

二、填空题

1. 答案：10300

解析：在 scanf 函数的格式说明符“%d%*d%d”中，第二个格式说明符为“%*d”，其作用跳过变量 j 的输入。所以整个输入语句是 i=10,j=30,k 没有被重新赋值，其值仍为初值 0。因而输出结果为 10300。

2. 答案：88

解析：以 0 开头的数是八进制数。所以 x=0210 表示八进制数 210，其二进制形式为 010001000。%x 表示以十六进制的形式输出，所以 010001000 的十六进制输出结果为 88。

3. 答案：123.460000

解析：printf 函数是 C 语言提供的标准输出函数，用来在终端设备上按指定格式进行输出。其调用形式如下：printf（格式控制，输出项 1，输出项 2，……），“格式控制”是字符串形式。格式说明符用%f 表示输出形式为浮点型输出，因此表达式(int)((x*100+0.5)/100.0)的输出结果为 123.460000。

4. 答案：10　11

解析：自变量 m 的值为八进制数 011，因为 011 的八进制数转换为十进制数后其值等于 9，然后在输出语句中变量 m 加 1 后输出其值 10，变量 n 是输出其值 11 后再加 1 为 12，所以输出结果是 10　11。

5. 答案：2 20.000000

解析：按照运算符优先级次序计算表达式 x=f*=n/=(c=50)的值，其运算次序为：1. 对 c 进行赋值；2. 计算表达式 n/=c 的值，即 n=n/c=100/50=2；3. 计算表达式 f*=n 的值，即 f=f*n=10.000000*2=20.000000；4. 将 f 的值赋给 x，即 x=20.000000。

第 8 章　习题分析与解答

一、选择题

1. 答案：D

解析：由 if else 语句的格式可知。

2. 答案：C

解析：C 语言中 if 之后的 scanf("%d",&x)没有分号，表示不是一个语句，所以出错。

3. 答案：C

解析：由 switch 语句的使用语法可知，在 switch 语句中可以根据需要使用 break 语句，另外，break 语句还可以用于 while，for 等循环语句。

4. 答案：A

解析：当 x 为 1 时，执行 case 1，a 自加等于 1，因为 case 1 后没有 break 语句，接着执行 case 2，此时 a 的值为 2，自加为 1。

5. 答案：A

解析：本题考查 switch 语句。当 i=1 时，执行 case 1，因为没有遇到 break 语句，所以依次往下运行，a=a+2=2，a=a+3=5；当 i=2 时，执行 case 2，因为没有遇到 break 语句，所以依次往下执行，a=a+2=7，a=a+3=10；当 i=3 时，执行 case 3，a=a+1=11，因为没有遇到 break 语句，所以依次往下执行，a=a+2=13，a=a+3=16；当 i=4 时，执行 default，a=a+3=19，结束循环。

6. 答案：①D　②C

解析：当输入 1 时，先与 case1 相匹配，一直执行到遇到 case4 的 break 语句。当输入 3 时，先与 case3 相匹配，一直执行到遇到 case4 的 break 语句。

7. 答案：C

解析：在 for 循环语句中自变量 i 从 0 开始，每次自加 2，执行 s+=*(i+1)语句，因此 C 语言规定数组名带数组的首地址，也就是第一个元素的地址，因此*(i+1)代表数组的第 i+1 个元素，所以程序运行的结果是 1+3+5+7+9=25，即变量 s 的值等于 25。

二、填空题

1. 答案：s=2，t=3

解析：在输入 5，2 后，a=5，b=2。执行语句①，s=2，执行语句②，a>b 为真，t=s+t=2+1=3。

2. 答案：y=0，y=5，y=10，y=5

解析：输入-10 时，x=-10，c=-1，执行 switch 语句，y=0；输入 5 时，x=5，执行 c=x/10=5/10=0，执行 switch 语句，y=x=5；输入 10 时，x=10，执行 c=x/10=10/10=1，

执行 switch 语句，y=10；输入 30 时，执行 c=x/10=30/10=3，执行 switch 语句，y=-0.5*x+20=5。

3. 答案：#&

解析：a=2，a>0 为真即为 1，执行外 switch 的 case1 语句，b=7，b<0 为假即为 0，不执行内 switch 的语句。转向外 switch 的 case0 语句，c=5，c=5 为真即为 1，执行内 switch 的 case1 语句，输出“#”，遇到 break 语句，退出内 switch 语句。转向外 switch 的 default 语句，输出“&”，退出外 switch 语句，输出“\n”。

4. 答案：3　2　2

解析：因为“b=a;a=c;c=b;”并没有用{ }括起来，因此，只有“b=a;”是 if 语句的子句，其他的赋值语句是独立的语句，不从属于 if 语句。a 中的值小于 c 中的值，因此不执行 if 后的“b=a;”，将直接执行“a=c;c=b;”结果，a 中的值变成 3，c 中的值变成 2，b 中值不变仍是 2。

三、程序设计题

```
1.#include "stdio.h"
   void main()
  {int m;
   scanf("%d",&m);
   if((m>0)&&(m%5==0)&&(m%7==0))
      printf("YES");
   else
     printf("NO");
 }
2.#include "stdio.h"
   void main()
  {int a,b,c,d,max;
   scanf("%d,%d,%d,%d",&a,&b,&c,&d);
   if(a>b)
     max=a;
   else
     max=b;
   if(max<c)
     c=max;
   if(max<d)
     d=max;
    printf("max is %d",max);
 }
3.#include "stdio.h"
  #include "math.h"
   void main()
 {  int x,y;
    scanf("%d",&x);
    if(x<-1)
      y=fabs(x);
    else if(x<1)
       y=2-sin(x);
         else if(x<=3)
             y=cos(x)+3;
             else
             y=x*x;
      printf("%d",y);
  }
```

第 9 章　习题分析与解答

一、选择题

1. 答案：A

解析：while 循环的执行过程如下：①计算 while 后面圆括号中表达式的值。当值为非 0 时，执行步骤②，当值为 0 时，执行步骤④；②执行循环体一次；③转去执行步骤①；④退出循环。在选项 A 中表达式(ch=getchar())!='N' 表示输入的字符不等于 N，如果这个条件表达式成立，则执行循环体，打印输出输入的字符；如果这个条件表达式不成立，即输入的字符等于 N，则退出循环。

2. 答案：D

解析：由 n!的数学定义可知 n!=n*(n-1)*(n-2)* ……*1。在选项 A 中，由于 f 的初值为 0，在 for 循环语句中，f 依次乘以 1，2，3，…，n，最后计算得到 f=n!=0，所以选项 A 不正确。在选项 B 中，f 的初值为 1，在 for 循环语句中，f 依次乘以 1，2，3，…，(n-1)，最后计算得到 f=(n-1)!，所以选项 B 不正确。在选项 C 中，f 的初值为 1，在 for 循环语句中，f 依次乘以 n，n+1，n+2，…，所以选项 C 不正确。在选项 D 中，f 的初值为 1，在 for 循环语句中，f 依次乘以 n，n-1，n-2，…，2，最后计算得到 f=n!，所以选项 D 正确。

3. 答案：B

解析：本题考查循环的使用，当 j=10，i=9 时，循环成立，第一个 if 条件为假，第二个 if 条件 i==j-1 成立则输出 j 的值为 10，接着执行 i++后 i 为 10，当 i 为 10，for 循环不成立，退出内层循环，此时执行 j++后，j 为 11，循环不成立，退出循环。

4. 答案：A

解析：当 k=5 时，while 条件中的值为 4，k 为 4，循环没有成立，k-=3 可写成 k=k-3，输出 1，接着执行--k 后 k 为 0，当 k=0 时，while 循环不成立。

5. 答案：A

解析：do while 的功能是先执行一次循环体，再判断条件是否成立，当 x=-1 时，执行 do 语句，x=x*x=1，接着执行 while 后的表达式，其值为 0，退出循环，因此循环体只执行了一次。

6. 答案：C

解析：for 循环结束时，i++的值应为 4，i 自加后此时应为 5。

7. 答案：B

解析：本题考查 do while 语句和 if else 语句。do while 语句的功能是先执行循环体再判断条件，所以，先判断 if 语句的条件，y=-4，!y 为逻辑 0，条件不成立，执行下面的 else 语句，输出 y，然后将 x 的值减 1，x=3 条件不成立，执行下面的 else 语句，输出 y，然后将 x 的值减 1，x=2，满足 while 循环条件，继续循环。第三次循环：先判断 if 语句的条件，y=2，!y 为逻辑 0，条件不成立，执行下面的 else 语句，输出 y，然后将 x 的值减 1，x=1，满足 while 循环条件，继续循环。第四次循环：先判断 if 语句的条件，y=-1，!y 为逻辑 0，条件不成立，执行下面的 else 语句，输出 y，然后将 x 的值减 1，x=0，不满足 while 循环条件，结束循环。

8. 答案：C

解析：当 if 执行到第一个满足(i*j>=20) &&(i*j<=100)这个条件的 i 出现时，通过 break 语句跳出循环，执行下面的 printf 语句。

9. 答案：B

解析：while 语句执行如下。

当 z=3，x=0 时，表达式 z-->0&&++x<5 为真，z 的值变为 2，x 的值变为 1，执行语句“y=y-1;”，y 的值变为 4；

当 z=2，x=1 时，表达式 z-->0&&++x<5 为真，z 的值变为 1，x 的值变为 2，执行语句“y=y-1;”，y 的值变为 3；

当 z=1，x=2 时，表达式 z-->0&&++x<5 为真，z 的值变为 0，x 的值变为 3，执行语句“y=y-1;”，y 的值变为 2；

当 z=0，x=3 时，表达式 z-->0 为假，表达式(z-->0&&++x<5)为假，z 的值变为-1，不进行++x<5 的判断，不执行语句“y=y-1;”。

10. 答案：C

解析：只有当 3 个 if 条件同时成立，即能够同时被 2，3，7 整除时，才输出 i 的值，而从 0 到 50 能够同时被 2，3，7 整除的数只有 42，被选择 C 选项。

11. 答案：B

解析：在第 1 次外层 for 循环中，首先 x++得到 x=1。进入到内层 for 循环，只有循环 j 的值为奇数时，变量的值才加 1，所以在内层 for 循环执行过程中，变量 x 的值自加两次，当退出内层 for 循环时，x=3，然后执行 x++，得到 x=4。再进入执行第 2 次外层 for 循环中，首先 x++得到 x=5。进入到内层 for 循环，只有循环变量 j 的值为奇数时，变量 x 的值才自加 1，所以在内层 for 循环执行过程中，变量 x 的值自加 1 两次，当退出内层 for 循环时，x=7，然后执行 x++，得到 x=8，所以打印输出变量 x 的值为 8。

二、填空题

1.答案：27

解析：此题就是求出 1 到 10 之，不是 3 的倍数的所有数之和。

2.答案：求出 1000 到 5000 之间千位与百位的和等于十位与个位的和

解析：本题主要是求出 4 位数的每一个数的位，然后通过程序，发现此程序的意图

3.答案：1,2

2,1

解析：通过本程序不难发现，求出的 i 和 j 的值之和是奇数。

三、程序设计题

```
1.#include "stdio.h"
  void main()
  {float sn=20,hn=2*sn/3;
   int n;
   for(n=2;n<=3;n++)
     {sn=sm+2*hn;
      hn=hn*2/3;
     }
   printf("第 3 次落地共经过%f 米\n",sn);
   printf("第 3 次反弹的%f 米\n",hn);
  }
2.#include <stdio.h>
  void main()
  {int i,j;
   for(i=1;i<=9;i++)
   {for(j=1;j<=i;j++)
      printf("%d*%d=%2d",i,j,i*j);
      printf("\n");
   }
}
3.#include "stdio.h"
  void main()
  {int cnt=0,sum=0,max=min=0;
   float v;
   while(1)
   {while(scanf("%d",&n),n<0);
     if(n==0)
       break;
     else
       {cnt++;
        sum=sum+n;
        if(n>max)
         max=n;
        if(n<min)
         min=n;
       }
   }
  v=(float)sum/cnt;
   printf("数据有%d 个，和是%d，平均值是%f，最大值是%d，最小值是%d",cnt,sum,v,max,min);
```

```
 }
4.#include "stdio.h"
  #include "math.h"
  void main()
  {float s=1,t=1,i=3;
   while(1)
    {if(i==3)
       t=i*(i-1)*(i-2);
     else
       t=t*i*(i-1);
     if(fabs(1.0/t)<1e-5)
       break;
    t=-t;
    s=s+1.0/t;
    i=i+2;
    }
    printf("s=%f",s);
   }
5.#include "stdio.h"
 void main()
 {int x,y,z,k=0;
  for(x=-45;x<=45;x++)
    for(y=-45;y<=45;y++)
      for(z=-45;z<=45;z++)
        if(x*x+y*y+z*z==2000)
         printf("x=%d,y=%d,z=%d\n",x,y,z);
 }
```

第 10 章　习题分析与解答

一、选择题

1. 答案：B

解析：选项 A 中对于二维数组，必须通过复制的个数来确定其长度，没有赋值则无法确定，选项 C 中赋的是空值，也不能确定其行下标，而选项 D 中定义的是两行三列数组，而赋值的却是三行两列数组。

2. 答案：B

解析：第 1 个循环的作用把从 0~9 的数赋给 a 数组；第 2 个循环的作用是把 a 数组中的部分元素放到数组 p 中，即 p[0]=a[0]=0，p[1]=a[2]=2，p[2]=a[6]=6；第 3 个循环的作用是把 p 中的各个元素的两倍之和放到 k 中，即 k=5+0+4+12=21。

3. 答案：A

解析：题中 while 循环的条件是：当从键盘读入的字符不是‘\n’时，执行 while 循环。

输入第 1 个字符 3 时：执行 c-'3'=0，执行 case0，什么也不输出，case 1，输出 7，case 2 输出 7，直到遇到 break 语句，跳出 switch 语句。

输入第 2 个字符 8 时：c-'3'=5，不执行任何语句。

输入第 3 个字符 4 时：c-'3'=1'，执行 case1，输入 8，case 2，输出 8，直到遇到 break 语句，跳出 switch 语句。

输入第 4 个字符 5 时：c-'3'=2'，执行 case2，输出 9，遇到 break 语句，跳出 switch 语句。

4. 答案：C

解析：语句“char p[ ]={'a','b','c'};”定义了一个一维字符数组 p[ ]，并用 3 个字符‘a’，‘b’，‘c’进行了初始化；而语句“q[ ]="abc";”表示定义了一个一维字符数组，并用一个字符串常量“abc”进行了初始化。在 C 语言中，系统在每个字符串常量的最后自动加上一个字符‘\0’作为字符串的结束符。所以函数 sizeof(q)=4，而 sizeof(p)=3。

5. 答案：C

解析：C 语言的字符型数据并不是将该字符本身存储到内存中，而是将该字符对应的 ASCII 码存储到内存单元中。

①原码：是一种机器数，最高位为 0 表示正数，为 1 表示负数，其余各位表示数本身；②反码：是对原码除符号位以外的各位按位求反得到的数；③补码：正数的补码和其原码相同，负数的补码最高位为 1，其余各位在原码的基础上按位求反，然后在最低位加 1 得到。

6. 答案：C

解析：首先，getchar 函数通过键盘读入字符‘a’，即 c=‘a’（其实 c 得到的是字符 a 的 ASCII 码值），然后判断 if 语句的控制条件，发现'a'>='a'&&'a'<='g'成立，则执行下面的语句，c=c+4，c 得到的是字符 e 的 ASCII 码值，退出 if 语句，通过 putchar 函数输出字符 e。

7. 答案：C

解析：数组名 s1 是代表 s1 数组首地址的地址常量，因为=左边不能出现常量，s1=“ABCDE”的方法是错误的。

二、填空题

1. 答案：92

解析：当 i=0 时，i 自动变为 1，if 语句不成立，执行后面的 do while 语句，将二维数组的第 2 行的 4 个元素累加到 s；当 i=1 时，i 自动变为 2，执行 if 语句，结束本次循环；当 i=2 时，i 自加变为 3，又不执行 if 而执行 do while 语句，将二维数组第 4 行的元素累加到 s；当 i=3 时，i 自动变为 4，执行 if 语句，退出本次循环；当 i=4 时，while 循环不成立。最后结果就是求数组 a 的第 2 行和第 4 行所有元素的和。

2. 答案：b[i]=a[i]+a[i+1]

解析：将数组 a 中元素 a[i]与 a[i+1]值相加后的值赋予数组 b 中元素 b[i]，即可实现将一个数组的前项和后项之和存入另一数组。

三、程序设计题

```
1.#include "stdio.h"
  void f(int a[],int n)
  {int i,j,temp;
   for(i=0;i<n-1;i++)
```

```
    for(j=0;j<n-1-i;j++)
     if(a[j]<a[j+1])
     {temp=a[j];
      a[j]=a[j+1];
      a[j+1]=temp;
     }
   }
  void main()
  {int a[10],i;
   for(i=0;i<10;i++)
      scanf("%d",&a[i]);
   f(a,10);
   for(i=0;i<10;i++)
      printf("%d",a[i]);
  }
2.#include "stdio.h"
  long f(int n)
  {long fact=1;
   int i,j;
   for(i=1;i<=n;i++)
     fact=fact*i;
   return fact;
  }
  void main()
  {int a,b;
   long s;
   scanf("%d%d",&a,&b);
   s=f(a)+f(b);
   printf("s=%ld",s);
  }
3.#include "stdio.h"
 long f(int m,int k)
  {int i,t=1;
   for(i=1;i<=k;i++)
      t=t*m;
   return t;
  }
  void main()
  {int i,n,k;
   long s=0;
   scanf("%d,%d",&n,&k);
   for(i=1;i<=n;i++)
      s=s+f(i,k);
   printf("s=%ld",s);
   }
```

第 11 章　习题分析与解答

一、选择题

1. 答案：D

解析：函数 f 的功能是对两个数据进行互换操作。在主函数中指针变量 p 和 q 分别指向数组 a[8]的首和尾，在 while 循环中实现从首尾开始对数组元素进行互换操作。

2. 答案：C

解析：由题目的已知条件函数 fun 的定义形式 void fun(char ch,float x){…}可知，函数 fun 定义为 void 型（无返回值型），故其没有返回值，所以选项 B 错误。函数 fun 的第 1 个形式参数为字符类型而不是字符串类型，所以选项 A 不符合要求。选项 D 的第一实参为整型与函数 fun 的第 1 个形式参数的类型不符，所以选项 D 也是错误的。选项 C 的第 1 个实参的类型为字符型、第 3 实参的类型为单精度型，与函数 fun 的形式参数的类型相符合，所以选项 C 为正确答案。

3. 答案：C

解析：子函数 fun1(double a)的功能是返回 a 的平方值的整数部分。子函数 fun2(double x,double y)的功能是返回 x 的平方值的整数部分与 y 的平方值的整数部分的和。又因为题中变量 w 的定义为 double 型，函数 fun2 定义为 int 型，按照各类数值型数据间的混合运算，整型数据被转换为实型数据。所以双精度型变量 w 的值为 5.0。

4. 答案：C

解析：函数 fun(int x,int y)的功能是返回两个整型数据的和。在主函数中，变量 a，b，c 的初始值分别为 1，2，3。因此逗号表达式“a++,b++,a+b”的值等于 5，表达式 c++的值为 3，调用子函数的表达式为“fun(5,3);”，其返回值等于 8。所以变量 sum 的值等于 8。

5. 答案：C

解析：因为在函数 fun(int x)中，如果参数 x 等于 0 或 1 时，返回值为 3。否则 p=x-fun(x-2)，这是一个递归函数，当在主函数中调用 fun(7)时，其过程为：“fun(7)=7-fun(5)=7-(5-fun(3))=7-(5-(3-fun(1)))=7-(5-(3-3))=7-5=2”，所以最后的输出结果为 2。

6. 答案：D

解析：本题考查函数实参和形参之间是怎样传递数值的。实参向形参传递数据的方式有两种：1. 如果实参传递给形参的是变量的值，进行的就是数值传递；2. 如果实参传递给形参的是变量的地址，进行的就是地址传递。

7. 答案：B

解析：本题考查库函数的调用。可以查看相关的库函数的调用形式。对于 tan(x)来说，它的调用形式为 tan(pi/x)。

8. 答案：B

解析：本题考查函数调用时的实参形式。这里含有对逗号运算的考查，(a,b)的运算结果为 b，因而只有一个值，结合 func 的调用形式，可以知道实参的个数只有两个。

9. 答案：B

解析：本题考查函数调用时的参数传递。在函数调用时，形参是指向实参的指针变量，则 prinf 的执行结果为 3+1=4。

10. 答案：A

解析：本题考查函数调用的基本概念。在函数调用时，只要符合函数的使用，程序中的各个函数间既可以直接调

用其他函数，也可以递归调用其自身。

11. 答案：C

解析：表达式 a++,b++,a+b 是一个逗号表达式，在逗号表达式中，从左向右进行各个表达式的运算，最后一个表达式的值就是逗号表达式的结果。所以表达式 a++,b++,a+b 的值为 5，所以 fun 函数中的形参 x 的值为 5；表达式 c++先把变量 c 的值传给形参 y，然后 c 的值加 1，所以 y 的值为 2(C++)，因此函数 fun((a++,b++,a+b),c++)的返回值为 7。

12. 答案：C

解析：本题考查 do while 循环。在 fun 函数中，首先定义了静态变量 i=0，do while 循环要实现的功能是实现 b[0]=b[0]+b[1]，b[1]=b[1]+b[2]，在其他元素不变的规则下重新给 b 数组赋值。在主函数中，通过调用 fun 函数，按上面的规则对数组 a 重新赋值，最后输出数组 a。

13. 答案：C

解析：在函数 fun(int x)中，有个 if 语句判断如果参数 x 等于 0 或 1 时返回值，否则进入下面的 p=x-fun(x-2)递归函数，当在主函数中调用 fun(7)时，其过程为："fun(7)=7-(5-fun(3))=7-(5-(3-fun(1)))=7-(5-(3-3))=7-5=2"，所以最后的输出结果为 2。

14. 答案：A

解析：在程序中当 i=0 时，s[0]=1，f(s[0])为 1，执行 d+=s[0]，d 的值为 1，i++也变为 1。当 i=1 时，s[1]=3，f(s[1])为 1，也为非 0，继续执行后面的表达式，此时 d 的值为 1+3=4，继续执行 i++变为 2，接着执行 for 循环，可知 for 循环的目的是把 s 数组中的奇数相加，因此最后 d 的值为 1+3+5 为 9。

15. 答案：A

解析：子函数 void change(int k[]) {k[0]=k[5];}的功能是用一维数组中的第 6 个元素替换第 1 个元素。因此主函数的 while 语句循环 5 次，每次循环都是用数组中当前指针所指元素后的 5 个元素替换该元素，因此退出 while 循环时，数组中前 5 个元素为 "6,7,8,9,10"。

16. 答案：A

解析：在函数 fun1()之前定义了全局字符变量 a 和 b。这两个变量的作用域是从其定义处开始到整个程序末结束。在函数 fun1()之内定义了两个变量 a 和 b，并且分别初始化为字符 'C' 和 'D'。C 语言中若全局变量和某个函数中的局部变量同名，则在该函数中，此全局变量被屏蔽，在该函数内，访问的是局部变量，与同名的全局变量不发生任何关系。所以在主函数中，执行 fun1()后，变量 a，b 的值分别等于 'C'，'D'，打印输出 CD，接着执行 fun2('E','F')语句，变量 a，b 的值变为 'E'，'F' 并输出。

17. 答案：B

解析：函数 void f(int b[])的功能是对数组 b[]中第 2 个到第 5 个元素的值逐个扩大 2 倍，所以在 main()中，f(a)语句的作用是对数组 a[10]中从 a[2]到 a[5]的各个数字乘以 2，因而数组 a[10]的元素就变成了{1,2,6,8,10,12,7,8,9,10}。

二、填空题

1. 答案：4 3 3 4

解析：在 C 语言中，数据从实参传递给形参，称为按值传递。也就是说，当简单变量作为实参时，用户不能在被调函数中改变实参的值。所以在本题主函数中，当执行调用函数 swap(a,b)时，将把实参 a 和 b 的值传递给形参 x 和 y，在函数 swap 中，形参 x 和 y 的值进行交换再输出，所以输出为 4 3，但主函数中实参 a 和 b 的值并没有改变，最后输出 a 和 b 的值为 4 3。因而程序输出结果为 4 3 3 4。

2. 答案：15

解析：函数f(int a[],int n)的作用是从后向前递归求出数组a[]中各个元素之和。所以主函数中的调用函数语句 s=f(aa,5)的执行过程如下：

```
s=f(aa,4)+aa[4] =f(aa,3)+aa[3]+aa[4]=f(aa,2)+ aa[3]+aa[4]=f[aa,1]+aa[1]+aa[2]+ aa[3]+ aa[4]= aa[0]+aa[1]+aa[2]+aa[3]+aa[4]=15。
```

3. 答案：8651

解析：本题考查程序输出，程序运行时，第 1 次循环，i=0 调用 sb(x,x)子函数，此时 n=3，x=s[n]=s[3]=8，n--，n 变为 2；第 2 次循环，i=1，调用 sb(s,x)子函数，因此将 n 定义为静态变量，所以此时 n=2，返回 x=s[n]=s[2]=6；第 3 次循环，i=2，调用 sb(s,x)子函数，此时 n=1，返回 x=s[n]=s[1]=5；第 4 次循环，i=3，调用 sb(s,x)子函数，此时 n=0，返回 x=s[n]=s[0]=1。

4. 答案： j+1　　i%2==1

解析：由程序段可知 i 表示行下标，j 表示列下标，所以第 1 个 for 循环表示逐行扫描，第 2 个 for 循环表示逐列扫描，所以 k 表示 i 行中从第 j 个元素之后的元素的列下标，因而变量 k 的初始值应该等于 j+1。条件 if 语句中的表达式为一个条件表达式，如果第 i 行为偶数行，则这一行的元素从小到大排序，否则这一行的元素从大到小排序。

5. 答案：010111212

解析：在 C 语言中，在函数中定义的静态局部变量，在程序退出函数时，其中的值不会释放。下次调用该函数时，静态局部变量的值仍然有效。由于在本题函数 fun()中的变量 c 为静态局部变量，所以调用 fun()函数结束后，变量 c 中的值不会释放，下次调用时，仍然有效。在主函数执行 for 循环过程中，当 i=0 时，fun(5)=5+1+4=10，此时 c=4；当 i=1 时，fun(5)=5+1+5=11，此时 c=5；当 i=2 时，fun(5)=5+1+6=12，此时 c=6；当 i=3 时，退出 for 循环，并输出回车换行符。因而输出结果为：010111212。

6. 答案：10

解析：函数 sub(int n)的作用是求得整数 n 的个位数字与其十分之一数字的和。在主函数中，对函数 sub(intn)进行了 3 次递归调用，当输入 1234 时，其执行顺序为：sub(1234)=127，sub(127)=19，sub(19)=10，所以结果为 10。

三、程序设计题

```
1.#include "stdio.h"
  void f(int a[],int n)
  {int i,j,temp;
   for(i=0;i<n-1;i++)
    for(j=0;j<n-1-i;j++)
     if(a[j]<a[j+1])
     {temp=a[j];
      a[j]=a[j+1];
      a[j+1]=temp;
     }
   }
  void main()
  {int a[10],i;
   for(i=0;i<10;i++)
   scanf("%d",&a[i]);
   f(a,10);
   for(i=0;i<10;i++)
   printf("%d",a[i]);
  }
2.#include "stdio.h"
  long f(int n)
  {long fact=1;
   int i,j;
   for(i=1;i<=n;i++)
     fact=fact*i;
   return fact;
  }
  void main()
  {int a,b;
   long s;
   scanf("%d%d",&a,&b);
   s=f(a)+f(b);
   printf("s=%ld",s);
  }
```

第 12 章　习题分析与解答

一、选择题

1. 答案：C

解析：本题考查 for 循环及 if 语句。当执行到第一个满足(i*i>=20)&&(i*i<=100)这个条件的 i 出现时，break 跳出循环，执行下面的 printf 语句。

2. 答案：A

解析：分析程序可知，函数中的参数在传递时传递的是指针 p，而不是 p 的地址，所以不会影响主函数中的指针 p，因而输出的字符是 b[0]（即为 a）。

3. 答案：A

解析：根据程序可知，函数 fun()的参数为指针类型，它的返回值为字符型。再看各选项，选项 B 和选项 C 中的参数不是指针类型，因此错误。选项 D 中的参数是指针类型，但它的返回值为指针型，因此也错误。

4. 答案：B

解析：在 int(*p)[3]中，圆括号中有“*”，表示 p 是一个指针，最后的说明符“[3]”表明是一个数组，前者优先，即 p 是指向含有 3 个元素的二维数组的行指针。

5. 答案：A

解析：该语句是合法的，“[]”的优先级高，说明 line 是一个数组，前面带有“*”，表示 line 数组的元素是 char 型指针。

6. 答案：B

解析：在子函数void f(char *s,char *t)中，字符指针s和t分别指向一个字符串的首部和尾部，从而实现字符串倒置的功能。在main函数中，由于strlen(str)2=3，所以p=str+4，即p指向str数组中下标为 4 的元素“e”，(p-2)指向str数组中下标为 2 的元素“c”，再调用f(p,p-2)函数后，实现把字符串str[10]的前半部分“abc”和后半部分“efg”分别从字符‘c’和‘e’开始实现两个字符串相应字符的互换。所以选项B正确。

7. 答案：C

解析：本题中 int (*ptr)()的含义是指向函数的指针变量，该函数返回一个 int 型数据。

8. 答案：C

解析：函数 void swap(char *x,char *y)的功能是交换两个字符*x 和*y 中的内容。在主函数中字符指针 s1 指向字符串“abc”，s2 指向字符串“123”。函数 swap(s1,s2)的执行结果就是字符‘a’和‘1’相互交换。所以选项 C 正确。

9. 答案：C

解析：程序先使整型指针 p1，p2，p3 指向 a，b，c 值所存放的地址空间，然后再将指针变量*p1，*p2 中的值的乘积赋给指针变量*p，即赋给变量 c，输出 c 的值 3。

10. 答案：A

解析：选项 A 表示在程序中，声明定义变量语句“int n, *p=NULL;”定义了整型变量 n 和指针变量 p，并且指针变量 p 初始化为空。根据题意选项 B 正确表示应该为“*p=n;”，选项 C 正确表示应该为“scanf("%d",p);”，选项 D 正确表示应该为“printf("%x\n",p);”。

11. 答案：D

解析：*号的优先级比+号的优先级高，所以先执行“*p”；指针 p 指向的是数组的首地址，因此*p=1，再加 8 等于 9。

12. 答案：A

解析：本题中 sub(int *s,int *y)函数的参数是两个指针型变量，在函数体内将数组 s 的第 1 个元素赋给 y。在主程序内，首先定义了一维数组并赋初值，然后通过 for 循环，5 次调用 sub(a,&x)函数，每一次调用都是将数组 a 的第 1 个元素赋给 x，并输出。

13. 答案：C

解析：本题考查的是指向数组的指针。函数指针的定义基本格式为：类型标识符（指针变量名）()。其中，类型标识符 p 是一个指针变量名，它指向一个含有 6 个字符型元素的一维数组。

14. 答案：B

解析：选项 A，是将指针 q1 和 q2 所指向的变量值相加然后赋给 k；选项 B 中，将 float 型数据和指针型数据之间进行赋值运算，这是不允许的；选项 C，是两个指针变量之间的赋值；选项 D，是两个指针型变量所指向的两个 float 型数据相乘。

15. 答案：C

解析：本题采用函数调用的方法，fun 中 if 语句成立则进行 i 和 j 的交换，用到主函数中则为把下标为 0、3，1、2 的元素互换，因此当执行完后 x 的数组为 8，1，6，2。

16. 答案：D

解析：在 point(char *p)函数中，只进行了值传递，没有进行地址传递，所以调用 point 函数后，变量 pt 的值未改变。

17. 答案：A

解析：本题考查字符串比较函数和两个字符串比较原则这两个知识点。

（1）两字符串比较的原则是一次比较两个字符串同一位置的一对字符，若它们的ASCⅡ码值相同，则继续比较下一对字符，若它们的ASCⅡ码不同，则ASCⅡ码值较大的字符所在的字符串较大；若所有字符相同，则两个字符串相等；若一个字符串全部i个字符与另一个字符串的前i个字符相同，则字符串较长的较大。（2）strcmp(s1,s2)的返回值，当str1<str2 时，返回值为负数；当str1=str2 时，返回值为 0；当str1>str2 时，返回值为正数。

18. 答案：A

解析：本题考查了函数之间地址值的传递，当形参为指针变量时，实参和形参之间的数据传递是地址传递，可以在被调用函数中对调用函数中的变量进行引用，在被调用函数中直接改变调用函数中变量的值，所以主函数中的 b 在带到 fun 函数后将发生变化，而 a 不发生变化，在调用子函数的时候输出 g 和 G，返回到主函数中输出的是 F 和 g。

19. 答案：B

解析：指针中存放的是变量的地址，指针也可以进行增减运算，这时候指针移动的最小单位是一个存储单元，而不是一个字节。所以题中 p+6 指的是将指针向后移动了 6 个存储单元，指向 b[6]，存放的是 b[6]的地址。

20. 答案：D

解析：*(q+i)指向第 i 行首地址，*(*(q+i)+j)代表第 i 行第 j 个元素。

21. 答案：D

解析：首先定义了一个指向一维数组 b 的指针 p，一个指向指针 p 的指针变量 q，输出指针 p 所指单元的内容，即 b[1]的值。

22. 答案：A

解析：本题考查的是指向函数的指针。函数指针的定义方式是：类型标识符（*指针 变量名）()。其中，“类型标识符”为函数返回值的类型，“指针”指向函数的入口地址。

二、填空题：

1. 答案：252H

解析：要解答本题，首先要明白在对指针进行加减运算时，数字 1 不是十进制数的 1，而是指 1 个存储单元长度。1 个存储单元长度占存储空间的多少，应该视具体情况而定。如果存储单元的基类型是 int 型，则移动 1 个存储单元的长度就位移 2 个字节；如果存储单元基类型是 float 型，则移动 1 个存储单元的长度就位移 4 个字节。所以 p+13 所指向的数组元素的地址：200H+(13*4)H=252H。

2. 答案：150

解析：本题先给变量 x 赋初始值 100，然后将指针 p 指向变量 x，*p 是取指针 p 所指地址的内容，即 100，所以 x=100+50=150。

3. 答案：ptr 是指向函数的指针，该函数返回一个 int 型数据。

解析：本题考查函数指针的概念。函数指针的定义格式是：类型标识符（*指针变量名）()。其中，“类型标识符”为函数返回值的类型。

4. 答案：*t

解析：首先用字符串函数 strlen(s)求出字符数组 s 的长度，然后 s+n 指向 s 数组中最后一个元素的后面，把 t 数组放在 s 之后，当遇到 t 数组的结束符“/0”时，while 判别式的值为 0，结束循环，即实现函数的功能。

5. 答案：*pmax=*px;

解析：把*px的值赋给*pmax，作为其初始值，然后其值再依次与*py，*pz的值相比较，最后求出 3 个数中的最大值。

6. 答案：68

解析：本题考查如何用指针引用数组元素。

本题先定义了一个指向字符型数组 str 的指针 p，指针 p 指向数组 str 的首地址，p+3 将指针指向 str[3], *(p+3)指的是字符‘D’，输出时是以“%d”格式输出的，即输出其相应的 ASCII 码值 68。

7. 答案：abcdelkjihgf

解析：本题先给字符型数组s的12个元素赋值a到l共12个字母，函数sub(char*a,int t1,int t2)的功能是将数组a从第t1+1个元素进行逆置，在主函数中调用sub(s,5, SIZE-1)函数，是将数组s的第6个元素到第12个元素进行逆置，其他元素位置不变。

8. 答案：a=8,b=7

解析：本题中 swap(p1,p2)函数的作用是交换两个形参的值，因为函数的形参是地址，形参和实参进行的是地址

传递，所以通过 swap 函数可以实现数据交换的目的。在主函数中，if 语句的控制条件 a<b 成立(a=7，b=8)，所以调用 swap 函数交换 a 和 b 的值。

三、程序设计题

```
1.#include "stdio.h"
  #inlcude "string.h"
  void main()
  {char a[80];
   int cnt=0,i;
   gets(a);
   for(i=0;s[i]!='\0';i++)
     cnt++;
   printf("%d",cnt);
   }
2.#include "stdio.h"
  #include "string.h"
  void main()
  {char b[100],a[100];
   int i,cnt=0,n,m;
   gets(a);
   scanf("%d",&m);
   n=streln(a);
   for(i=0;i<=n-m;i++)
     b[cnt++]=a[m-1+i];
   b[cnt]='\0';
   puts(b);
  }
3.#include "stdio.h"
  #include "string.h"
  void main()
 {char *temp,**p;
  char *nation[5]={"Amercia","China","Korea",
  "Japanese","Aferica"};
  int i,j,k;
  p=nation;
  for(i=0;i<4;i++)
    for(j=0;j<4-i;j++)
     if(strcmp(*(p+j),*(p+j+1))>0)
      {temp=*(p+j);
       *(p+j)=*(p+j+1);
       *(p+j+1)=temp;
      }
   for(i=0;i<5;i++)
   printf("%s\n",*(p+i));
 }
```

第 13 章　习题分析与解答

一、选择题

1. 答案：D

解题：本题考查宏替换的规则。宏替换分为简单的字符替换和带参数的宏替换两类。使用宏时应注意以下几点：1. 宏定义仅仅是符号替换，不是赋值语句，因此不做语法检查；2. 为了区别程序中其他的表示法，宏名的定义通常用大写字母，但不是必须用大写；3. 双引号中出现的宏名不替换；4. 使用宏定义可以嵌套，即后定义的宏中可以使用先定义的宏。

2. 答案：C

解析 带参数宏的格式为：#define 标识符（形参表）形参表达式。其功能是：预处理程序将程序中出现的所有带实参的宏名，展开成由实参组成的表达式。

3. 答案：C

解析：本题考查文件包含的知识点。格式 1：#include，<文件名>，预处理程序在标准目录下查找指定的文件；格式 2：#include “文件名”，预处理程序首先在引用被包含文件的源文件所在的目录中寻找指定的文件，如没找到，再按系统指定的标准目录查找。

4. 答案：D

解析：预处理命令行包括宏命令，是在预编译阶段即正式编译之前进行处理的。

5. 答案：C

解析：由于预处理命令行是在预编译阶段而不是程序执行时进行处理的，所以宏替换不占运行时间，只占编译时间。

6. 答案：C

解析：本题考查宏的使用规则：1.字符替换格式为“#define 标识符 字符”，其中“标识符”称为宏名，为类型；2.双引号中出现的宏名不替换；3.宏名的定义通常用大写字母，但不是必须用大写；4 宏定义不是赋值语句，不做语法检查。

7. 答案：C

解析：本题考查带参数的宏的运算，当宏定义为#define f(x) x*x 时，c=4*4/2*2=16；若宏的定义改为#define f(x) (x*x)，则 c=(4*4)/(2*2)=4，要注意区分。

8. 答案：B

解析：本题考查带参数的宏的定义及其相关运算，s=m+n+m+n*k=1+2+1+2*3=10。

9. 答案：B

解析：z=2*(N+Y(5))=2*(2+((2+1)*5))=2*(2+15)=34。

二、填空题

1. 答案：8

解析：本题考查带参数的宏的定义及其相关运算。运算过程为：t=B*2=A+3*2=2+3*2=8。

2. 答案 16.000000

解析：本题考查带参数的宏定义及其相关运算。其运算过程为：c=B(a)=4*8.0/2=16.000000。

3. 答案：the end

解析：本题考查带参数的宏定义。第 1 次循环，i=1，

调用 PRINT(1)，P(1)输出 1；第 2 次循环，i=2，调用 PRINT(2)，P(2) 输出 2；第 3 次循环，i=3，调用 PRINT(3)，P(3)输出 3；第 4 次循环，i=4，调用 PRINT(4)，P(4)输出 4，the end。

4. 答案：字符串

解析：本题考查字符替换格式：#define 标识符 字符串。

5. 答案：55

解析：分析程序执行过程，第 1 次循环时，j=3，i=5，因为 switch(3)，所以执行 case3，调用 p(a[--i])=p(a[4])=p(5)，输出 5；第 2 次循环时，j=2，i=4，因为 switch(2)，所以执行 case2，调用 p(a[i++])=p(a[4])=p(5)，输出 5 之后 i 自加等于 5。

第 14 章　习题分析与解答

一、选择题

1. 答案：A

解析：本题主要考查结构体数组的使用。x[i].num 为结构体 x[i]中的 num 成员，x[i].name[2]是结构体 x[i]中 name 成员的第 3 个元素。程序执行循环过程如下：第 1 次循环，i=1，输出 x[1].num，x[i]name[2]，即 2A；第 2 次循环，i=2，输出 x[2].num，x[2].name[2]，即 3N；第 3 次循环，i=3，输出 x[3].num，x[3].name[2]的值，即 4A；第 4 次循环，i=4，输出 x[4].num，x[4].name[2]的值，即 5U。

2. 答案：C

解析：本题主要考查结构体数组。a[2].age 为结构体 a[2]的 age 成员，即 16。a[3].name 为指向结构体 a[3]的 name 成员的第 1 个元素的指针，即指向“Z”，（a[3].name+2）将指针后移两位指向第 3 个元素“A”，*(a[3].name+2)是取指针所指向地址的内容。

3. 答案　A

解析：题中 a1 和 a2 两个结构体变量名所对应的成员相同，可以与运算符“++”相结合，结构体变量的输出格式为：printf（“要输出变量名：%d/t，结构变量名 要输出的成员变量名”）。

4. 答案：C

解析：本题主要考查结构指针。p=&st 访问结构体的成员，可以通过结构变量访问，即 st.i，也可以用等价的指针形式：(*p).i 和 p->i。

5. 答案：C

解析 ：结构体变量 person 的 birthday 又是一个结构体变量，对 person 的出生年份的引用应是 person. birthday.year。

6. 答案：D

解析：程序中进行了两个宏定义，即#define HDY(A,B)和#define PRINT(Y) printf("y=%d/n,y")，所以根据宏替换规则，在主函数中 K=HDY(a+c,b+d)=1+3，2+4=6。因此本题的答案为选项 D。

7. 答案：C

解析：在位运算中，操作数每右移两位，相当于操作数的左侧添 0，最右面的两位被移出，即操作数除以 4。

8. 答案：A

解析：逻辑位运算的特定作用主要有3点：1. 用按位与运算将特定位清为0或保留特定位；2. 用按位或运算将特定位置为1；3.用按位异或运算将某个变量的特定位翻转或交换两个变量的值。

9. 答案：B

解析：本题主要考查结构指针：p=&data，访问结构体的成员。它可以通过结构变量访问，即 data，可以用等价的指针形式：(*p).a 和 p–>a 来访问结构体变量中的成员。

10. 答案：B

解析：从实参传递过去的是结构体系数组的第 3 个元素，所以输出的 name 为 Zhao。

二、填空题

1. 答案：4

解析：由于主函数中“s[0].next=s+1;s[1].next=s+2;s[2].next=s;”，可知结构体 NODE 数组 S[3]形成一个循环队列。于是有*p=s[0]，*q=s[1]，*r=s[2]，sum+=q->next->num=s[2].num3，因而 sum=0+3=3，sum+=r->next->next->num=s[1].num=1，所以 sum=3+1=4。

三、程序设计题

```
1. #include "stdio.h"
   struct data{
   int y,m,d;}
   void main()
   {  int s;
      struct data a;
      scanf("%d-%d-%d",&a.y,&a.m,&a.d);
      s=a.d;
     switch(a.m-1)
    {   case 11:s+=30;
        case 10:s+=31;
        case 9:s+=30;
        case 8:s+=31;
        case 7:s+=31;
        case 6:s+=30;
        case 5:s+=31;
        case 4:s+=30;
        case 3:s+=31;
        case 2:if(a.y%4==0&&a.y%100!=0||a.
                  y%400==0)
        s+=29;
        else
        s+=28;
        case 1:s+=31;
    }
```

```
    printf("%d-%d-%d 是该年的第%d 天\n",a.
    y,a.d,s);
  }
2. #include "stdio.h"
   struct std_info
   {  char no[10];
      char name[9];
      float score;
   }a[5];
   void main()
   {   float aver=0,max;
       int i,t;
       for(i=0;i<5;i++)
       {  scanf("%s",a[i].no);
          scanf("%s",a[i].name);
          scanf("%f",a[i].score);
       }
       max=a[0].score;
       t=0;
       for(i=0;i<5;i++)
       { aver=aver+a[i].score;
         if(max<a[i].score)
         { max=a[i].score;
           t=i;
         }
       }
      printf("aver=%f",aver);
      printf("no:%s\n",a[t].no);
      printf("name:%s\n",a[t].name);
      printf("score:%f\n",a[t].score);
  }
3. stuct std_info *edit(struct std_info
   *head,char no[],float score)
     {struct std_info *p;
      if(head==NULL)
        return NULL;
      p=head;
      while(p!=NULL&&strcmp(p->no,no))
        p=p->next;
      if(p!=NULL)
        p->score=score;
      else
        printf("学号%s 找不到!\n",no);
      return head;
     }
```

第 15 章　习题分析与解答

一、选择题

1. 答案：B

解析：本题考查的内容是按位与和按位或运算符的使用，由运算规则可知，d 的最终结果为 4。

2. 答案：B

解析：^为按二进制异或运算符，变量 s 与其相等的数值异或的结果为 0。

3. 答案：B

解析：x 对应的二进制数为 00000011，y 对应的二进制数为 00000110，移位运算符的优先级高，y<<2 的结果是 00011000，所以最终结果为 00011011。

4. 答案：A

解析：本题考查对位运算知识的理解与掌握。注意整数要转化为二进制，然后来进行位运算。左移运算符<<是双目运算符，其功能把<<左边的运算数的各二进位全部左移若干位，由<<右边的数指定移动的位数，高位丢弃，低位补 0。

5. 答案：C

解析：本题主要考查左移运算。4 对应的二进制数为 00000100，左移一位，右端补一个 0，结果为 00001000，对应的十进制数为 8。可见，每左移一位，相当于移位对象乘以 2。

6. 答案：B

解析：位段结构体变量 a 的总长度为 24 个位，即 3 个字节。

7. 答案：D

解析：在 16 位机上，位段的最大长度为 16，本题定义的位段长度为 20，所以编译有错。

第 16 章　习题分析与解答

一、选择题

1. 答案：D

解析：字符串输入函数 fgets 的调用形式为：“fgets(s;n;fp);”。

fgets 函数的功能是：从 fp 所指向的文件中读取长度不超过 n-1 个字符的字符串，并将该字符串放到字符数组 s 中；读入字符串后会自动在字符串末尾加入结束符，表示字符串结束。

2. 答案：A

解析：位置指针当前值函数 ftell()的基本调用格式为：ftell(fp)。ftell 函数的功能是：得到 fp 所指向文件的当前读写位置，即位置指针的当前值，如果函数的返回值为-1L，表示出错。

3. 答案：C

解析：fread 函数的调用形式为：“fread(buffer,size,count,fp);”。

其中，“buffer”是一个指针，对 fread 来说，它是读入数据的存放地址，对 fwrite 来说，是要输出数据的地址；“size”是要读写的字节数；“count”是要进行读写多少个

size 字节的数据项；“fp”是指文件型指针。

4. 答案：B

解析：用“w”方式打开文件意思是如果文件名相同则覆盖原来的文件，所以当再次输入时覆盖了第一次的内容，选择 B 选项。

5. 答案：C

解析：“PRN”是打印机设备文件名，“fp=fopen("PRN", "w");”语句的功能是打开打印机，向其中写内容即打印内容。

6. 答案：D

解析：在 C 语言中，有两种对文件的存取方式：顺序存取和直接存取；如果以“a”方式对一个已打开的文件进行写操作后，则原有文件中内容将保存，新的数据写在原有内容之后。如果以“a+”方式为读和写操作打开一个文件，则既可以对文件进行读，也可以对文件进行写，而且在读和写操作之间不必关闭文件，可以从头开始读。当对文件的读（写）操作完成之后，必须将它关闭。

7. 答案：D

解析：使用文件函数 fopen 打开 d2.dat 文件并把数组 a[6]中的元素分两行写入到 d2.dat 文件中，关闭文件。然后再次打开文件 d2.dat，用 fscanf 函数读取文件 d2.dat 中的数据，因为每行没有分隔符，所以每一行会被认为是一个完整的数，并存入到变量 k 和 n 中，因此输出变量 k 和 n 的值为 123　456。

8. 答案：B

解析：本题考查库函数调用的知识点。格式 1：#include <文件名>，预处理程序在标准目录下查找指定的文件；格式 2：#include “文件名”，预处理程序首先在引用被包含文件的源文件所在的目录中寻找指定的文件，如没找到，再按系统指定的标准目录查找。

9. 答案：A

解析：位置指针当前值函数 ftell 的基本调用格式为：“ftell(fp);”。

ftell 函数的功能：得到 fp 所指向文件的当前读写位置，即位置指针的当前值，如果函数的返回值为-1L，表示出错。

10. 答案：D

解析：本题考查 feof(fp)函数，其功能是：测试所指文件的位置指针是否已达到文件尾，如果已达到文件尾，则函数返回非 0 值；否则返回 0，表示文件结束。

11. 答案：B

解析：改变文件位置的指针函数 fseek 的调用形式为，“fseek(fp,offset,position)”。

其中，“fp”是指向该文件的文件型指针；“offset”为位移量，指从起始点位置到要确定的新位置的字节数，也就是以起点为基准，向前移动的字节数。

二、填空题

1. 答案："a"

解析：“a”表示以向文本文件尾增加数据的方式打一个文件。

2. 答案："rb"

解析：“rb”表示以只读方式打开一个二进制文件。通过循环给 p 赋值，可以依次向后取结点的值。

三、程序设计题

1. 答案：

假设 C 盘下已经建立了 f1 文件。

```
#include "stdio.h"
#include "string.h"
void main()
{  FILE *fp;
   char str[10];
   int i;
   gets(str);
   if((fp=fopen("c:\\f1.txt","w"))==NULL)
  {  printf("can not open f!\n");
     exit(0);
  }
   for(i=;i<streln(str);i++)
      str[i]=str[i]+32;
      fputs(str,fp);
      fclose(fp);
}
```

2. 答案：

假设 C 盘下有文件 c。

```
#include "stdio.h"
#inlcude "string.h"
void main()
{  FILE *fp;
   char *s1="abcdefg",*s2="12345";
   int i;
   char s3[80];
   if((fp=fopen("c:\\c.txt","w"))==NULL)
   {  printf("can not open f!\n");
    exit(0);
   }
  for(i=0;i<3;i++)
    s3[i]=s1[i];
  s3[i]='\0';
  strcat(s2,s3);
  fputs(s2,fp);
  fclose(fp);
}
```

附录 B　2010 年 3 月二级笔试试卷

2010 年 3 月全国计算机等级考试二级笔试试卷

C 语言程序设计

（考试时间 90 分钟，满分 100 分）

一、选择题（（1）～（10）、（21）～（40）每题 2 分，（11）～（20）每题 1 分，共 70 分）下列各题 A）、B）、C）、D）4 个选项中，只有一个选项是正确的。请将正确选项填涂在答题卡相应位置上，答在试卷上不得分。

（1）下列叙述中正确的是（　　）。

A）对长度为 n 的有序链表进行查找，最坏情况下需要的比较次数为 n

B）对长度为 n 的有序链表进行对分查找，最坏情况下需要的比较次数为（n/2）

C）对长度为 n 的有序链表进行对分查找，最坏情况下需要的比较次数为（$\log_2 n$）

D）对长度为 n 的有序链表进行对分查找，最坏情况下需要的比较次数为（$n\log_2 n$）

（2）算法的时间复杂度是指（　　）。

A）算法的计算时间

B）算法所处理的数据量

C）算法程序中的语句或指令条数

D）算法在执行过程中所需要的基本运算次数

（3）软件按功能可以分为：应用软件、系统软件和支持软件（或工具软件）。下面属于系统软件的是（　　）。

A）编辑软件　　B）操作系统

C）教务管理系统　　D）浏览器

（4）软件（程序）调试的任务是（　　）。

A）诊断和改正程序中的错误

B）尽可能多地发现程序中的错误

C）发现并改正程序中的所有错误

D）确定程序中错误的性质

（5）数据流程图（DFD 图）是（　　）。

A）软件概要设计的工具

B）软件详细设计的工具

C）结构化方法的需求分析工具

D）面向对象方法的需求分析工具

（6）软件生命周期可分为定义阶段、开发阶段和维护阶段。详细设计属于（　　）。

A）定义阶段　　B）开发阶段

C）维护阶段　　D）上述 3 个阶段

（7）数据库管理系统中负责数据模式定义的语言是（　　）。

A）数据定义语言　　B）数据管理语言

C）数据操纵语言　　D）数据控制语言

（8）在学生管理的关系数据库中，存取一个学生信息的数据单位是（　　）。

A）文件　　B）数据库

C）字段　　D）记录

（9）数据库设计中，用 E-R 图来描述信息结构但不涉及信息在计算机中的表示。它属于数据库设计的（　　）。

A）需求分析阶段　　B）逻辑设计阶段

C）概念设计阶段　　D）物理设计阶段

（10）有两个关系 R 和 T 如下：

R

A	B	C
a	1	2
b	2	2
c	3	2
d	3	2

T

A	B	C
c	3	2
d	3	2

则由关系 R 得到关系 T 的操作是（　　）。

A）选择　B）投影　C）交　D）并

（11）以下叙述正确的是（　　）。

A）C 语言程序是由过程和函数组成的

B）C 语言函数可以嵌套调用，例如 fun(fun(X))

C）C 语言函数不可以单独编译

D）C 语言中除了 main 函数，其他函数不可作为单独文件形式存在

（12）以下关于 C 语言的叙述中正确的是（　　）。

A）C 语言的注释不可以夹在变量名或关键字的中间

B）C 语言中的变量可以在使用之前的任何位置进行定义

C）在 C 语言算术表达式的书写中，运算符两侧的运算数类型必须一致

D）C 语言的数值常量中夹带空格不影响常量值的正确表示

（13）以下 C 语言用户标识符中，不合法的是（　　）。

A）_1　B）AaBc　C）a_b　D）a--b

（14）若有定义：“double a=22; int i=0, k=18;”，则不符合 C 语言规定的赋值语句是（　　）。

A）a=a++,i++;　　B）i=(a+k)<=(i+k）;

C）i= a %1;　　D）i=!a;

（15）有以下程序：

```
#include <stdio.h>
main ( )
{ char a,b,c,d;
  scanf("%c%c",&a&b);
  c=getchar();   d=getchar();
  printf ("%c%c%c%c\n",a,b,c,d);
}
```

当执行程序时，按下列方式输入数据（从第 1 列开始，<CR>代表回车，注意：回车也是一个字符）。

12<CR>

34<CR>

则输出结果是（　　）。

A）1234　B）12　　C）12　　D）12

3　　34

（16）以下关于 C 语言数据类型使用的叙述中错误的是（　　）。

A）若要准确无误差地表示自然数，应使用整数类型

B）若要保存多少位的小数的数据，应使用双精度类型

C）若要处理如“人员@”有不知类型的相关数据，应自定义结构体类型

D）若只处理“真”与“假”两种逻辑值，应使用逻辑类型

（17）若 a 是数值类型，则逻辑表达式(a==1)||a<=1)的值是（　　）。

A）1　　B）0

C）2　　D）不知道 a 的值，不能确定

（18）以下选项中与“if(a==1)a=b;else a++;”语句功能不同的 switch 语句是（　　）。

A）switch(a)
{case1:a=b;break;
default: a++;
}

B）switch(a==1)
{case0:a=b；break;
case1:a++;
}

C）switch(a)
{default:a++;break;
case1:a=b;
}

D）switch(a==1)
(case1:a=b;break;
case0 : a++;
}

（19）有如下嵌套的 if 语句（　　）。

```
if(a<b)
if(a<c)  k=a;
else  k=c;
else
if(b<c) k=b;
else  k=c;
```

以下选项中与上述 if 语句等价的语句是（　　）。

A）k=(a<b)?a:b;k=(b<c)?b:c;

B）k=(a<b)?((b<c)?a:b):((b>c)?b:c);

C）k=(a<b)?((a<c)?a:b):((b<c)?b:c);

D）k=(a<b)?a:b;k=(a<c)?a:c;

（20）有以下程序：

```
#include  <stdio.h>
main()
{int i,j,m=1;
 for(i=1;i<3;i++)
 { for(j=3;j>0;j--)
   {if(i*j>3) break;
      m*n=i*j;
   }
 }
```

```
    printf("m=%d\n",m);
}
```

程序运行后的输出结果是（　　）。

A）m=6　　B）m=2

C）m=4　　D）m=5

（21）有以下程序：

```
#include  <stdio.h>
main()
{ int a=1,b=2;
  for(;a<8;a++) {b==a;a+2;}
    printf("%d,%d\n",a,b);
```

程序运行后的输出结果是（　　）。

A）9,18　　B）8,11

C）7,11　　D）10,14

（22）有以下程序，其中 k 的初值为八进制数。

```
#include "stdio.h"
main()
( int k=011;
  printf("%d\n",k++);
}
```

程序运行后的输出结果是（　　）。

A）12　　B）11

C）10　　D）9

（23）下列语句组中，正确的是（　　）。

A）char *s;s="Olympic";

B）char s[7];s="Olympic";

C）char *s;s={"Olympic"};

D）char s[7];s={"Olympic"};

（24）以下关于 return 语句的叙述中正确的是（　　）。

A）一个自定义的函数中必须有一条 return 语句

B）一个自定义的函数中可以根据不同情况设置多条 return 语句

C）定义成 void 类型的函数中可以有带返回值的 return 语句

D）没有 return 语句的自定义函数在执行结束时不能返回到调用处

（25）下列选项中，能正确定义数组的语句是（　　）。

A）int num(0..2008);　　B）int　num=1;

C）int num2008;　　D）#define N 2008 int num[N]

（26）有以下程序：

```
#include <stdio.h>
 void  fun(char *c,int d)
{ *c=*c*i; d=d+1;
  print("&c,&c,",*c,d);
}
main()
{ char b='a',a='A';
  fun(&b,a);printf("&c,&c\n",b,a);
}
```

程序运行后的输出结果是（　　）。

A）b, B ,b , A　　B）b, B ,B , A

C）a, B, B, d　　D）a, B, a, B

（27）若有定义 int (*pt)[3]，则下列说法正确的是（　　）。

A）定义了基类型为 int 的 3 个指针变量

B）定义了基类型为 int 的具有 3 个元素的指针数组 pt

C）定义了一个名为*pt、具有 3 个元素的整型数组

D）定义了一个名为 pt 的指针变量，它可以指向每行有 3 个整数元素的三维数组

（28）设有定义 double a[10]，*s=a，以下能够代表数组元素 a[3]的是（　　）。

A）(*s)[3]　　B）(*s+3)

C）*s[3]　　D）*s+3

（29）有以下程序：

```
#include <stdio.h>
main()
{ int a[5]={1,2,3,4,5},b[5]={0,2,1,3,0}, i,
  s=0;
  for (i=0;i<5;i++)   s=s+a[b[i]];
    printf("&d\n",s);
}
```

程序运行后的输出结果是（　　）。

A）6　　B）10　　C）11　　D）15

（30）有以下程序：

```
 #include  <stdio.h>
main()
{ int b[3][3]={0,1,2,0,1,2,0.1.2},i,j,t=1;
  for(i=0;i<3;i++)
    for (j=i;j<=i;j++)
      printf("&d\n",t);
}
```

程序运行后的输出结果是（　　）。

A）1　　B）3　　C）4　　D）9

（31）若有以下定义和语句：

```
{ar s1[10]="abcd",*s2="\n123\";
printf("%d %d\n",strlen(s1),strlen(s2)}
```

则输出结果是（　　）。

A）5　5　　B）10　5　　C）10　7　　D）5　8

（32）有以下程序：

```
#include <stdio.h>
#define x i
void fun(int *x,int i)
{ *x=*(x+i);}
 main()
 {int a[N]={1,2,3,4,5,6,7,8},i;
  fun(a,2);
  for(i=0;i<N/2;i++)
    {printf("&d",a[i];)}
  printf("\n");
}
```

程序运行后的输出结果是（　　）。

A）1313　　B）2234

C）3234　　D）1234

（33）有以下程序：

```
#include <stdio.h>
int f(int t[],int n);
 main()
 {int a[4]=(1,2,3,4),s;
  str(s,c); printf("%d\n",s);
}
int f(int t[],int n)
{if (n>0) return t(n-1)+f(t,n-1);
 else  return  0};
}
```

程序运行后的输出结果是（　　）。

A）4　　B）10　　C）14　　D）6

（34）有以下程序：

```
 #include <stdio.h>
 int fun()
{static int x=1;
 x*2; return x;
}
main()
{ int i,s=1;
  for(i=1;i<=2;i++)
    printf("%d\n",s);
}
```

程序运行后的输出结果是（　　）。

A）0　　B）1　　C）4　　D）8

（35）有以下程序：

```
#include <stdio.h>
#definc sub(a)  (a)={a}
 main()
{ int a=2,b=3,c=5,d;
  d=sub(a+b)*c;
  printf("%d\n",d);
}
```

程序运行后的输出结果是（　　）。

A）0　　B）-12　　C）-20　　D）10

（36）设有定义：

```
   srtuct complex
 {  int resl,unreal;} data!={1,8},data2;
```

则以下赋值语句中错误的是（　　）。

A）data2=data1;　　B）data2={2,6}

C）data2.real=data1.real;　　D）data2.real=data1.unreal;

（37）有以下程序：

```
  #include  <stdio.h>
  #include  <string,h>
   struct   A
{  int a; char b[10]; double c;};
   void f(struct   A  t);
 main()
 { struct A  a={1001,"ZhangDa",1098,0};
   f(a); printf ("%d,%s,%6.1f\n",a.a,a.b,a.c);
}
void f(struct At)
{  t.a=1008;  strcpy(t.b"ChangRong");  t.c=
1202.0;}
```

程序运行后的输出结果是（　　）。

A）1001,ZhangDa,1098.0

B）1002,ChangRong,1202.0

C）1001,ChangRong,1098.0

D）1002,ZhangDa,1202.0

（38）有以下定义和语句。

```
  struct   workers
   {int num;char  name[20]; char c;
struct
```

```
{int  day; int month;  int year;}s;
};
struct workers  w, *pw;
pw=&w;
```

能给 w 中 year 成员赋 1980 的语句是（　　）。

A）*pw.year=1980

B）w.year=1980

C）pw->year=1980

D）w.s.year=1980

（39）有以下程序：

```
#include <stdio.h>
main()
{ int a=2,b=2,c=2;
  printf("%d\n",s/b&c);
  }
```

程序运行后的输出结果是（　　）。

A）0　　B）1　　C）2　　D）3

（40）有以下程序：

```
#include <stdio.h>
 main()
 {FILE *fp;char str[10];
  fp=fopen("myfile,dat","w");
  fopen("abc",i<>1,fclose(fp));
  fopen("myfile,dat","a");
  fprintf(fp,"%d",28);
  fewind(fp);
  fscanf(fp," %s",str); puts(str);
  fclose(fp);
}
```

程序运行后的输出结果是（　　）。

A）abc　　B）28c

C）abc28　　D）3 类型不一致输出错

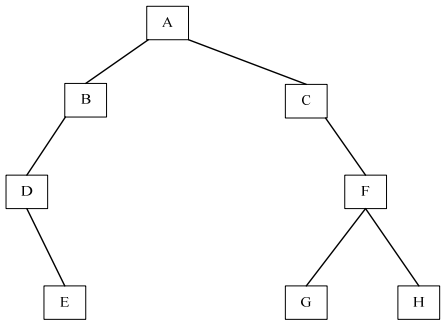
二、填空题（每空 2 分，共 30 分）

请将每空的正确答案写在答题卡【1】～【15】序号的横线上，答在试卷上不得分。

（1）一个队列的初始状态为空，现将元素 A，B，C，D，E，F，5，4，3，2，1 依次入队，然后再依次退队，则元素排列的顺序为【1】。

（2）设某循环队列的容量为 50，如果头指针 front= 45，<指向队头元素的前一位置>，尾指针 real=10（指向队尾元素），则该循环队列中共有【2】个元素。

（3）设二叉树如下：



对该二叉树进行后序遍历的结果为【3】。

（4）软件是【4】、数据和文档的集合。

（5）有一个学生选课的关系，其中学生的关系模式为：学生（学号、姓名、班级、年龄），课程的关系模式为：课程（学号、课程名、学时），其中两个关系模式的键分别是学号和课程号，则关系模式选课可定义为：选课（学号、【5】、成绩）。

（6）设 x 为 int 型变量，请写出一个表达式【6】，用以判断 x 同时为 3 和 7 的倍数时，关系表达式的值为真。

（7）有以下程序：

```
#include <stdio.h>
main()
{int a=1,b=2,c=3,d=0;
  if(a==1)
   if(b==2)
    if(c==3)     d=1;
    else          d=2;
   else if(c!=3)       d=3;
        else         d=4;
   else             d=5;
   printf ("%d\n", d );
      }
```

程序运行后的输出结果是【7】。

（8）有以下程序：

```
#include <stdio.h>
main()
{ int m,n;
  scanf("%d%d",&n,&m);
  while(m!=n)
    {while(m>n) m=1,-n;
     while(m<n) n=1,-m
    }
  printf("%d\n", m);
      }
```

程序运行后，当输入 14 63<回车>时，输出结果是【8】。

（9）有以下程序：

```
#include  <stdio.h>
main()
{  int i,j,a[][3]={1,2,3,4,5,6,7,8,9};
   for (i=0,i<3,i++)
     for (j=i,j<3,j++)
       printf ("%d",a[i][j]);
   printf ("\n");}
```

程序运行后的输出结果是【9】。

（10）有以下程序：

```
#include <stdio.h>
main()
{  int a[]={1,2,3,4,5,6},*k[3],i=0;
   while(i<3)
   {k[i]=&a[2*i];
   printf("%d",z8k(i));i++;}}
```

程序运行的输出结果是【10】。

（11）有以下程序：

```
#include <stdio.h>
main()
{  int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};
   int b[2][0],i;
   for(i=0;i<3;i++) b[i]=a[i][2]+a[2][i];
   for(i=0;i<3;i++)   printf("%d",b[i]);
   printf("\n");}
```

程序运行的输出结果是【11】

（12）有以下程序：

```
#include <stdio.h>
#include <string.h>
void fun(char *str)
{  char temp;int i,n;
   N=strlen(str);
   temp=str(n-1);
   for(i=n-1;i>0;i--)
   str[i]=str(i-1);
     str[0]=temp;}
main()
{  char s[50];
   scanf("%s",s); fun(s);printf("%s\n",s);}
```

程序运行后输入：abcdef< 回车>，则输出结果是【12】。

（13）以下程序的功能是：将值为 3 位整数的变量 x 中的数值按照个位、十位、百位的顺序拆分并输出。请填空。

```
#include <stdio.h>
main()
{  int x=256;
   printf("%d%d%d\n", 【13】,x/10%10,x/100;)}
```

（14）以下程序用以删除字符串中所有的空格，请填空。

```
#include <stdio.h>
main()
{ char s[100]="our teacher teach c language!";
  int i,j;
  for (i=j=0;s[i]!='\0';i++)
     if (s[i]!=' ')  s[j]=s[i];j++
         s[j]= 【14】;
  printf ("%s\n",s);}
```

（15）以下程序的功能是：借助指针变量找出数组元素中的最大值及元素的下标值，请填空。

```
#include  <stdio.h>
main()
{  int a[10],*p,*s;
   for(p=a;p-a<10;p++)  scanf("%d",p);
   for(p=a,s=a;p-a<10;p++)if(*p>*s)  s=【15】;
   printf("index=%d\n",s-a);
}
```

附录 C　2010 年 3 月份试卷分析

一、选择题

（1）答案：A

解析：最坏情况为倒序排列，因此需要从头到尾查找一遍，比较次数为 n。

（2）答案：D

解析：算法在执行过程中所需要的基本运算次数为算法实际执行时间的估量。

（3）答案：B

解析：编辑软件、教务管理系统和浏览器均属于应用软件，只有操作系统属于系统软件。

（4）答案：A

解析：调试过程一般用于确认错误发生的位置，查找出导致错误的原因并修改。

（5）答案：C

解析：DFD 数据流程图是面向过程的分析工具，且不是概要和详细设计的工具。

（6）答案：B

解析：首先 D 可以排除，维护阶段系统已经完成，不可能再进行设计，而在定义阶段业务需求尚未清楚，只能进行概要设计。

（7）答案：A

解析：数据操纵语言 DML（Data Manipulation Language），用户通过它可以实现对数据库的基本操作，例如，对表中数据的查询、插入、删除和修改；数据控制语言 DCL（Data Control Language）是用来设置或者更改数据库用户或角色权限的语句，这些语句包括 GRANT、DENY、REVOKE 等；无数据管理语言的说法。

（8）答案：D

解析：记录包含学生信息的各个字段内容。文件和数据库是数据容器，存放所有学生的信息。

（9）答案：A

解析：E-R 图为实体关系图，在需求分析阶段使用。

（10）答案：B

解析：关系运算包括：选择、投影、连接等。选择又称为限制，它是在关系 R 中选择满足给定条件的诸元组；在关系 R 上的投影是从 R 中选择出若干属性列组成新的关系；由关系 R 和 T 的内容，并不存在交与并的关系。

（11）答案 B

解析：根据 C 语言定义可以知道，A 错，因为 C 语言是由函数组成，C 错可以单独编译，D 显然错误。

（12）答案：B

解析：根据 C 语言中有关定义说明，A 错注释可以在任何地方，B 正确，C 错不用一致也可以进行运算，D 带空格的话不能正确地表示数值，顾也错。

（13）答案：D

解析：此题考查 C 语言标识符。

①第一个字符必须是字母（不分大小写）或下画线（_）。

②后跟字母（不分大小写）、下画线（_）或数字。

③标识符中的大小写字母有区别。如，变量 sum、SUM、Sum 代表 3 个不同的变量。

④不能与 C 编译系统已经预定义的、具有特殊用途的保留标识符（即关键字）同名。比如，不能将标识符命名为 float、auto、break、case、this、try、for、while、int、char、short、unsigned，等等。

根据以上可以得知本题应该选 D。

（14）答案：C

解析：C 项赋值表达式右侧计算结果是双精度类型，存储空间长度大于左侧的整型类型，因此不允许这样的赋值。

（15）答案：C

解析：本题主要要求掌握函数 getchar()的功能，即从隐含的输入设备输入一个字符，如果是多个输入，只有第一个有效。

（16）答案：D

解析：此题考查数据类型。

数据类型分：基本类型、构造类型、指针类型、空类型。

基本类型里又包括整型、字符型、浮点型、枚举型。

构造类型里又包括数组、结构体、共用体。

D 符合题目要求所以选 D，因为逻辑类型不属于数据类型。

（17）答案：A

解析：此题要注意||，如果||左边的表达式成立的话是不会执行||后面的表达式的，只要这个式子有一个表达式的结果为真的时候，结果就为真，而 a==1 和 a!=1 这两个式子，无论 a 的值是什么，整个式子的结果都为 1，所以此题答案为 1。选 A。

（18）答案：B

解析：此题考查 if 和 switch 之间的转化问题，A 中如果 a 不为 0 则执行 casel，与题目中给出的 if else 语句相同。同理不难看出 C 和 D 都是正确的。而 B 是错的，因为它的意思是当表达式为 0 时是不会执行 switch 里面的语句的。所以选 B。

（19）答案：C

解析：此题考查条件语句，做这种题目要仔细地看 if 和 else 的配对问题。else 始终与前面最近的未配对的 if 配对，还要看清条件，然后看选项，选项中的三目运算符是由右向左进行运算的，所以不难看出选 C。

（20）答案：A

解析：此题要注意有双重循环，要看清楚程序中的判断语句，推敲不难得出答案。

（21）答案：D

解析：此题中要注意循环，循环结束的条件也要注意，特别是后面的 a++和 a+=2，程序每执行一次，a 的值变化 3。

（22）答案：D

解析：此题中先把 k 化为十进制等于 9 ，这里可能你会感觉很不明白，既然是 9，那输出为什么还是 9 呢？对，这里还要注意 k++，是先取出 k 的值再自增 1 的。

（23）答案：A

解析：此题考查字符串的赋值，要注意数组名是不能进行字符串的直接赋值的，顾选项 B，D 都是错误的，还要注意的是指针的赋值不需要大括号。

（24）答案：B

解析：此题考查的是 return 语句。return 在 C 语言中的函数中出现，而且可以多条语句出现，所以 A 错，C 中当定义成 void 是不能出现 return 语句的，D 也错，应该是返回不确定的值。

（25）答案：D

解析：此题考查数组的定义，形式是：数值类型 数组名[下标]所以 A 错不能有()。B 也错，不是数组定义。C 也错没有[ ]。

（26）答案：A

解析：此题考查指针和函数调用。程序从 main 函数开始，调用函数，执行子函数，输出了 b 和 B，然后回到 main 函数输出了 b 和 A，所以答案是 A。

（27）答案：D

解析：此题考查行指针的定义，(*pt)[3]表示指针指向了一个由 3 个元素组成的一维数组。

（28）答案：B

解析：此题考查指针的引用。A 表示指向有 3 个元素组成的一维数组，B 正确，C 表示指针所以错，D 相当于 a[0]+3，显然错误。

（29）答案：C

解析：此题关键要注意 a[b[i]]，很显然 a 数组把 b 数组的值当做了下标，知道这点后就不难发现答案。

（30）答案：C

解析：考查二维数组。代码比较简单。

（31）答案：A

解析：此题考查 strlen 函数的用法，它的作用是计算出字符串中的字节数，所以 s1 的值为 5，特别要注意 s2，其中的特殊字符只占一个字节，所以字节数是 5，顾选 A。

（32）答案：C

解析：考查函数调用，先开始就进行了函数调用，所以数组中的值已经发生改变，不难发现答案选 C。

（33）答案：B

解析：此题考查函数调用中的递归调用，返回值为 10。

（34）答案：C

解析：此题要注意，static 表示静态定义，先自动清零改变后的值不发生改变。所以仔细看程序也不难得出答案。

（35）答案：C

解析：宏定义，当执行到宏定义是代入#define 可得到答案。

（36）答案：B

解析：此题为结构体。根据宏定义的规定其中 A，C，D 都是合法的，而 B 的方法是错误的。

（37）答案：A

解析：考查结构体，通过函数调用再通过 strcpy 函数进行复制，再对 t 进行赋值，其他的不变原样输出，顾为 A。

（38）答案：D

解析：本题主要是考查结构体的有关知识点，本题中的结构体中又包含了结构体，要使结构体中的 year 的值 1980，就必须能够使用定义的结构体变量去引用 year，如果使用指针变量 pw，那么用(pw->s).year 或者 w.s.year 来表示，所以，答案选 D。

（39）答案：A

解析：此题考查位运算，要注意&的前面不成立的话是不会执行&后面的。

（40）答案：C

解析：本题首先执行 fputs("abc",fp)，将 abc 写入到文件指针 fp 指定的文件里，然后执行 fp=fopen("myfile. data", "a++")，表示在文件 myfile.data 的文件末尾添加数据，而执行 fprintf(fp,"%d",28)表示把 28 添加到 fp 指针所指的文件的尾部。所以答案为 abc28。

二、填空题

（1）答案：ABCDEF54321

分析：队列的特点是先进先出，后进后出。

（2）答案：15

分析：在循环队列中，如果队头指针小于队尾指针，那队列中的元素为队尾减去队头，即为循环队列中的元素个数，如果队头指针大于对尾指针，那么队列中的元素为队尾+（容量−队头指针）=10+(50−45)=15。

（3）答案：EDBFHFCA，

分析：根据后序的特性。

（4）答案：程序

分析：软件的定义。

（5）答案：课号

分析：根据数据库中的关系模式的定义可填。

（6）答案：(x%3==0)&&(x%7==0)

分析：此题考查条件的选定。

（7）答案：4

分析：这题要好好理解，要注意 else 总是和最近的而且还没有配对过的 if 配对，仔细分析就会发现规律。

（8）答案：7

分析：此题考查循环，输入 14 和 63，赋给了 m 和 n，然后再判断条件是否满足，满足就进入循环，循环里面还有循环这是难点，但是只要注意分析就会得出答案。

（9）答案：123569

分析：此题考查循环的输出，关键是第 2 个循环的启始条件是从 i 开始的，注意到这里也就差不多能得出结论了。

（10）答案：135

分析：此题中既有循环又有指针，它定义了一个指针数组，分别指向 a[0]，a[2]，a[4]，所以只要输出这 3 个就可以了。

（11）答案：101418

分析：此题考查二维数组的运算和输出，关键是第 1 个循环中的运算，明白就可以得到最后输出的答案。

（12）答案：12fabcde

分析：此题的关键是子函数中的运算。

（13）答案：x%100%10

分析：此题就是 3 位数中各位数字的求法。

（14）答案：‘\0’

分析：此题要求删除空格，所以遇到空格就把它覆盖掉，最后以‘\0’结束。

（15）答案：p

分析：此题应用指针找出数组中最大的值的下标，第 1 个循环用来输入，而第 2 个用指针指向的值进行比较，在比较的过程中对 p 进行赋值。

附录 D　2010 年 9 月笔试试卷及解析

2010 年 9 月全国计算机等级考试二级笔试试卷

C 语言

（考试时间 90 分钟，满分 100 分）

一、选择题（（1）～（10）、（21）～（40）每小题 2 分，（11）～（20）每小题 1 分，共 70 分）

下列各题（A）、（B）、（C）、（D）4 个选项中，只有一个选项是正确的。请将正确选项填涂在答题卡相应位置上，答在试卷上不得分。

1．下列叙述中正确的是（　　）。

A）线性表的链式存储结构与顺序存储结构所需要的存储空间是相同的

B）线性表的链式存储结构所需要的存储空间一般要多于顺序存储结构

C）线性表的链式存储结构所需要的存储空间一般要少于顺序存储结构

D）以上 3 种说法都不正确

2．下列叙述中正确的是（　　）。

A）在栈中，栈中元素随栈底指针与栈顶指针的变化而动态变化

B）在栈中，栈顶指针不变，栈中元素随栈底指针的变化而动态变化

C）在栈中，栈底指针不变，栈中元素随栈顶指针的变化而动态变化

D）上述 3 种说法都不对

3．软件测试的目的是（　　）。

A）评估软件可靠性　B）发现并改正程序中的错误

C）改正程序中的错误　D）发现程序中的错误

4．下面描述中，不属于软件危机表现的是（　　）。

A）软件过程不规范　　B）软件开发生产率低

C）软件质量难以控制　　D）软件成本不断提高

5．软件生命周期是指（　　）。

A）软件产品从提出、实现、使用维护到停止使用退役的过程

B）软件从需求分析、设计、实现到测试完成的过程

C）软件的开发过程

D）软件的运行维护过程

6．面向对象方法中，继承是指（　　）。

A）一组对象所具有的相似性质

B）一个对象具有另一个对象的性质

C）各对象之间的共同性质

D）类之间共享属性和操作的机制

7．层次型、网状型和关系型数据库划分原则是（　　）。

A）记录长度　　B）文件的大小

C）联系的复杂程度　　D）数据之间的联系方式

8．一个工作人员可以使用多台计算机，而一台计算机可被多个人使用，则实体工作人员与实体计算机之间的联系是（　　）。

A）一对一　　B）一对多

C）多对多　　D）多对一

9．数据库设计中反映用户对数据要求的模式是（　　）。

A）内模式　　B）概念模式

C）外模式　　D）设计模式

10．有 3 个关系 R，S 和 T 如下：

R

A	B	C
a	1	2
b	2	1
c	3	1

S

A	D
c	4

T

A	B	C	D
c	3	1	4

则由关系 R 和 S 得到关系 T 的操作是（　　）。

A）自然连接　　B）交

C）投影　　D）并

11．以下关于结构化程序设计的叙述中正确的是（　　）。

A）一个结构化程序必须同时由顺序、分支、循环 3 种结构组成

B）结构化程序使用 goto 语句会很便捷

C）在 C 语言中，程序的模块化是利用函数实现的

D）由 3 种基本结构构成的程序只能解决小规模的问题

12．以下关于简单程序设计的步骤和顺序的说法中正确的是（　　）。

A）确定算法后，整理并写出文档，最后进行编码和上机调试

B）首先确定数据结构，然后确定算法，再编码，并上机调试，最后整理文档

C）先编码和上机调试，在编码过程中确定算法和数据结构，最后整理文档

D）先写好文档，再根据文档进行编码和上机调试，最后确定算法和数据结构

13．以下叙述中错误的是（　　）。

A）C 程序在运行过程中所有计算都以二进制方式进行

B）C 程序在运行过程中所有计算都以十进制方式进行

C）所有 C 程序都需要编译链接无误后才能运行

D）C 程序中整型变量只能存放整数，实型变量只能存放浮点数

14．有以下定义：int a; long b; double x,y; 则以下选项中正确的表达式是（　　）。

A）a%(int)(x-y)　　B）a=x!=y;

C）(a*y)%b　　D）y=x+y=x

15．以下选项中能表示合法常量的是（　　）。

A）整数：1,200　　B）实数：1.5E2.0

C）字符斜杠：‘\’　　D）字符串：“\007”

16．表达式 a+=a-=a=9 的值是（　　）。

A）9　　B）-9

C）18　　D）0

17．若变量已正确定义，在“if (W)printf("%d\n,k");”中，以下不可替代 W 的是（　　）。

A）a＜＞b+c　　B）ch=getchar（）

C）a==b+c　　D）a++

18．有以下程序：

```
#include <stdio.h>
main()
{int  a=1,b=0;
  if(!a)  b++;
  else  if(a==0)if(a)b+=2;
        else  b+=3;
  printf("%d\n",b);
}
```

程序运行后的输出结果是（　　）。

A）0　　B）1

C）2　　D）3

19．若有定义语句“int a,b;double x;”，则下列选项中没有错误的是（　　）。

A）switch(x%2)
{ case0: a++; break;
case1: b++; break;
default:a++; b++;
}

B）switch((int)x/2.0)
{ case0: a++; break;
case1: b++; break;
default:a++; b++;
}

C）switch((int)x%2)
{ case0: a++; break;
case1: b++; break;
default : a++; b++;
}

D）switch((int)(x)%2)
{ case0.0: a++; break;
case1.0: b++; break;
default :a++; b++;
}

20．有以下程序：

```
#include <stdio.h>
main( )
  { int a=1,b=2;
    while(a<6){b+=a;a+=2;b%=10;}
    printf("%d,%d\n",a,b);
  }
```

程序运行后的输出结果是（　　）。

A）5,11　　B）7,1

C）7,11　　D）6,1

21．有以下程序：

```
#include <stdio. h>
main( )
  { int y=10;
    while(y--);
    printf("y=%d\n",y);
  }
```

程序执行后的输出结果是（　　）。

A）y=0　　B）y= -1

C）y=1　　D）while 构成无限循环

22．有以下程序：

```
#include <stdio.h>
main( )
  { char s[]="rstuv";
    printf("%c\n",*s+2);
  }
```

程序运行后的输出结果是（　　）。

A）tuv　　B）字符 t 的 ASCII 码值

C）t　　D）出错

23．有以下程序:

```
#include  <stdio.h>
#include  <string.h>
main( )
  { char x[]="STRING";
    x[0]=0; x[1]='\0'; x[2]='0';
    printf("%d  %d\n",sizeof(x),strlen(x));
  }
```

程序运行后的输出结果是（　　）。

A）6　1　　B）7　0

C）6　3　　D）7　1

24．有以下程序:

```
#include <stdio. h>
int  f(int  x);
main( )
 { int  n=1,m;
   m=f(f(f(n)));
   printf("%d\n",m);
 }
int  f(int  x)
  {return  x*2;}
```

程序运行后的输出结果是（　　）。

A）1　　B）2

C）4　　D）8

25．以下程序段完全正确的是（　　）。

A）int *p; scanf("%d",&p);

B）int *p; scanf("%d",p);

C）int k, *p=&k; scanf("%d",p);

D）int k, *p; *p=&k; scanf("%d",p);

26．有定义语句：“int *p[4];”，以下选项中与此语句等价的是（　　）。

A）int　p[4];　　B）int　**p;

C）int　*(p[4]);　　D）int　(*p)[4];

27．下列定义数组的语句中，正确的是（　　）。

A）int　N=10;
　　int　x[N];　　B）#define N 10
　　int x[N];

C）int　x[0..10];　　D）int x[];

28．若要定义一个具有 5 个元素的整型数组，以下错误的定义语句是（　　）。

A）int a[5]=｛0｝;　　B）int b[]={0,0,0,0,0};

C）int c[2+3];　　D）int i=5,d[i];

29．有以下程序：

```
#include <stdio.h>
void  f(int *p);
main( )
  { int  a[5]={1,2,3,4,5},*r=a;
    f(r); printf("%d\n",*r);
  }
void f(int *p)
   { p=p+3; printf("%d, ",*p);}
```

程序运行后的输出结果是（　　）。

A）1,4　　B）4,4

C）3,1　　D）4,1

30．有以下程序（函数 fun 只对下标为偶数的元素进行操作）：

```
#include <stdio.h>
void fun(int *a,int n)
  { int i,j,k,t;
    for(i=0;i<n-1;i+=2)
      { k=i;
        for(j=i;j<n;j+=2)
        if(a[j]>a[k]) k=j;
        t=a[i];a[i]=a[k];a[k]=t;
      }
    }
main( )
   { int aa[10]={1,2,3,4,5,6,7},i;
     fun(aa,7);
     for(i=0;i<7;i++)
     printf("%d,",aa[i]);
     printf("\n");
    }
```

程序运行后的输出结果是（　　）。

A）7,2,5,4,3,6,1,　　B）1,6,3,4,5,2,7,

C）7,6,5,4,3,2,1,　　D）1,7,3,5,6,2,1,

31．下列选项中，能够满足“若字符串 s1 等于字符串 s2，则执行 ST”要求的是（　　）。

A）if(strcmp(s2,s1)==0) ST;　　B）if(sl==s2) ST;

C）if(strcpy(sl,s2)==1) ST;　　D）if(sl-s2==0) ST;

32．以下不能将 s 所指字符串正确复制到 t 所指存储空间的是（　　）。

A）while(*t=*s) {t++;s++;}

B）for(i=0;t[i]=s[i];i++);

C）do {*t++=*s++;} while(*s);

D）for(i=0,j=0;t[i++]=s[j++];);

33．有以下程序（strcat 函数用以连接两个字符串）：

```
#include  <stdio.h>
#include  <string.h>
main( )
 { char a[20]="BCD\OEFG\0",b[]="IJK";
   strcat(a,b);
   printf("%s\n",a);
 }
```

程序运行后的输出结果是（　　）。

A）ABCDE\OFG\OIJK　　B）ABCDIJK

C）IJK　　D）EFGIJK

34．有以下程序，程序中库函数 islower (ch)用以判断 ch 中的字母是否为小写字母。

```
#include  <stdio.h>
#include  <ctype.h>
void  fun(char *p)
  { int  i=0;
    while (p[i])
     { if(p[i]==' '&&islower(p[i-1]))
       p[i-1]= p[i-1]-'a'+'A';
       i++;
      }
  }
main( )
   { char s1[100]="ab cd EFG ! ";
     fun(s1); printf("%s\n",s1);
   }
```

程序运行后的输出结果是（　　）。

A）ab　cd　EFG ！　　B）Ab　Cd　Efg ！

C）aB　cD　EFG ！　　D）ab cd Efg ！

35．有以下程序：

```
#include  <stdio.h>
void  fun(int x)
  { if(x/2>1) fun(x/2);
    printf("%d",x);
  }
main()
  { fun(7); printf("\n");}
```

程序运行后的输出结果是（　　）。

A）1 3 7　　B）7 3 1

C）7 3　　D）3 7

36．有以下程序：

```
#include  <stdio.h>
int fun( )
   { static int x=1;
     x+=1;return x;
   }
main()
   { int i;s=1;
     for(i=1;i<=5;i++) s+=fun();
     printf("%d\n",s);
   }
```

程序运行后的输出结果是（　　）。

A）11　　B）21　　C）6　　D）120

37．有以下程序：

```
#inctude  <stdio.h>
#include  <stdlib.h>
main()
   { int *a,*b,*c;
     a=b=c=(int*)malloc(sizeof(int));
     *a=1;*b=2;*c=3;
     a=b;
     printf("%d,%d,%d\n",*a,*b,*c);
   }
```

程序运行后的输出结果是（　　）。

A）3,3,3　　B）2,2,3

C）1,2,3　　D）1,1,3

38．有以下程序：

```
#include  <stdio.h>
main()
{ int s,t,A=10;double B=6;
  s=sizeof(A);t=sizeof(B);
  printf("%d,%d\n",s,t);
}
```

在 VC 6.0 平台上编译运行，程序运行后的输出结果是（　　）。

A）2,4　　B）4,4

C）4,8　　D）10,6

39．若有以下语句：

```
typedef struct S
{int g; char h;} T;
```

以下叙述中正确的是（　　）。

A）可用 S 定义结构体变量

B）可用 T 定义结构体变量

C）S 是 struct 类型的变量

D）T 是 struct S 类型的变量

40．有以下程序：

```
#include <stdio. h>
main()
 { short c=124;
   c=c_______;
   printf("%d\n",C);
 }
```

若要使程序的运行结果为 248，应在下画线处填入的是（　　）。

A）>>2　　B）|248

C）&0248　　D）<<1

二、填空题（每空 2 分，共 30 分）

请将每空的正确答案写在答题卡【1】～【15】序号的横线上，答在试卷上不得分。

1．一个栈的初始状态为空。首先将元素 5，4，3，2，1 依次入栈，然后退栈一次，再将元素 A，B，C，D 依次入栈，之后将所有元素全部退栈，则所有元素退栈（包括中间退栈的元素）的顺序为【1】。

2．在长度为 n 的线性表中，寻找最大项至少需要比较【2】次。

3．一棵二叉树有 10 个度为 1 的结点，7 个度为 2 的结点，则该二叉树共有【3】个结点。

4．仅由顺序、选择（分支）和重复（循环）结构构成的程序是【4】程序。

5．数据库设计的 4 个阶段是：需求分析，概念设计，逻辑设计，【5】。

6．以下程序运行后的输出结果是【6】。

```
#include <stdio.h>
main()
  { int a=200,b=010;
    printf("%d%d\n",a,b);
  }
```

7．有以下程序：

```
#include <stdioh>
main()
  { int x,y;
    scanf("%2d%ld",&x,&y);
    printf("%d\n",x+y);
  }
```

程序运行时输入：1234567 程序的运行结果是【7】。

8．在 C 语言中，当表达式值为 0 时表示逻辑值“假”，当表达式值为【8】时，表示逻辑值“真”。

9．有以下程序：

```
#include <stdio. h>
main()
  { int i,n[]={0,0,0,0,0};
    for(i=1;i<=4;i++)
    { n[i]=n[i-1]*3+1;
      printf("%d ",n[i]);
    }
}
```

程序运行后的输出结果是【9】。

10．以下 fun 函数的功能是：找出具有 N 个元素的一维数组中的最小值，并作为函数值返回。请填空（设 N 已定义）。

```
int fun(int x[N])
 { int i,k=0;
   for(i=0;i<N;i++)
     if(x[i]<x[k]) k=【10】;
   return x[k];
 }
```

11．有以下程序：

```
#include <stdio.h>
int *f(int *p,int *q);
main( )
   { int m=1,n=2,*r=&m;
     r=f(r,&n); printf("%d\n",*r);
     }
  int *f(int *p,int *q)
      { return(*p>*q)?p:q;}
```

程序运行后的输出结果是【11】。

12．以下 fun 函数的功能是在 N 行 M 列的整型二维数组中，选出一个最大值作为函数值返回，请填空（设 M 和 N 已定义）。

```
int fun（int a[N][M]）
   {  int i,j,row=0,col=0;
      for（i=0;i<N;i++）
        for（j=0;j<M;j++）
          if(a[i][j]>a[row][col]){row=i;col=j;}
        return(【12】);
   }
```

13．有以下程序：

```
#include <stdio.h>
main()
```

```
{ int  n[2],i,j;
   for(i=0;i<2;i++) n[i]=0;
   for(i=0;i＜2;i++)
    for(j=0;j<2;j++) n[j]=n[i]+1;
   printf("%d\n",n[1]);
 }
```

程序运行后的输出结果是　【13】　。

14. 以下程序的功能是：借助指针变量找出数组元素中最大值所在的位置并输出该最大值。请在输出语句中填写代表最大值的输出项。

```
#include  <stdio. h>
main()
  { int a[10],*p,*s;
    for(p=a;p-a<10;p++) scanf("%d",p);
    for(p=a,s=a;p-a<10;p++) if(*p>*s) s=P;
    printf("max=%d\n",　【14】　);
 }
```

15. 以下程序打开新文件 f.txt，并调用字符输出函数将 a 数组中的字符写入其中，请填空。

```
#include <stdio.h>
main()
  {　【15】　*fp;
   char a[5]={'1','2','3','4','5'},i;
    fp=fopen("f.txt",s"w");
    for(i=0;i<5;i++)  fputc(a[i],fp);
    fclose(fp);
  }
```

试题解析

一、选择题

1.【解析】本题主要考查线性表的顺序存储结构和链式存储结构，顺序存储结构是线性表的一种最常用的存储方式。它用一组地址连续的存储单元来存储线性表的元素，用地址的相邻性来反映元素的逻辑关系；而链式存储结构可以采用不相连的存储单元来存储线性表的元素，它采用链表结点中指针域的值来连接各元素。链式存储时，存储一个元素一般需要两倍或多倍于顺序存储所需要的存储空间，不过链式存储对于线性表的插入和删除操作特别方便。

【答案】B

2.【解析】栈是限定仅在一端进行插入和删除操作的线性表。允许插入和删除的一端叫做栈顶，另一端叫做栈底，因此栈中元素随栈顶指针的变化而动态变化，而栈底指针不变。

【答案】C

3.【解析】软件测试的目的是尽可能多地发现程序中的错误，而不是为了单纯的改正程序中的错误。在本题中很多同学可能会觉得答案 B 会更合适，但事实上对软件测试目的描述最恰当的是选项 D。

【答案】D

4.【解析】在软件开发中遇到的问题找不到解决办法，使问题积累起来，形成了尖锐的矛盾，因而导致了软件危机。软件危机表现在以下几个方面：

（1）经费预算经常突破，完成时间一再拖延。

（2）开发的软件不能满足用户要求。

（3）开发的软件可维护性差。

（4）开发的软件可靠性差。

（5）软件开发费用不断增加。

（6）软件开发生产效率低下。

【答案】A

5.【解析】软件生命周期是指从软件定义、开发、使用、维护到报废为止的整个过程，一般包括问题定义、可行性分析、需求分析、总体设计、详细设计、编码、测试和维护等阶段。

【答案】A

6.【解析】继承性指的是一个新类可以从现有的类中派生出来，新类具有父类中所有的特性，直接继承了父类的操作和属性，同时也允许多个新类继承一个父类，也可以实现多层继承，可以说继承是类之间共享属性和操作的机制。

【答案】D

7.【解析】层次模型、网状模型和关系模型是目前数据库中最常用的 3 种数据模型，划分它们的原则是数据之间的联系方式。层次模型用树形结构来表示各实体与实体间的联系，而网状模型用网状结构来表示各实体与实体间的联系，而关系模型用表格形式表示实体类型及其实体间的联系。

【答案】D

8.【解析】在本题中，题目告诉我们一个工作人员可以使用多台计算机，而一台计算机可以被多个人使用，因此工作人员与计算机之间的联系应该是多对多的联系。

【答案】C

9.【解析】本题主要考查数据库设计的 3 种模式。数据库设计的 3 种模式分别是内模式、概念模式（模式）和外模式。其中概念模式是对数据库中数据的整体逻辑结构和特征的描述，是对所有用户的数据进行综合抽象而得到的统一的全局数据视图；外模式是对各个用户或程序所涉及的数据的逻辑结构和数据特征的描述，是完全按用户自己对数据的需要，站在局部的角度进行设计的；内模式是对数据的内部表示或底层描述。

【答案】C

10.【解析】本题主要考查关系运算。题目要我们根据关系 T 和 S 得出关系 T，其中关系 R 有 3 个属性，关系 S 有 2 个属性，而关系 T 有 4 个属性，如果对关系运算熟悉的话，我们不难推断出是自然连接运算。自然连接就是将 2 个具有相同属性的关系连接在一起，将两个关系中所有不同的属性都写在新的关系中。

【答案】A

11.【解析】结构化程序是利用函数来实现的，结构化程序可以是由顺序、分支、循环 3 种结构组成，但并不一定要求每个结构化程序都包含这 3 种结构，而且在结构化程序中，对于 goto 语句的使用都是有严格限制的，不能随意使用。当然结构化程序不仅仅局限于解决小规模的问题，只要合理利用，也能解决较大规模的问题。

【答案】C

12.【解析】本题主要考查程序设计的步骤。简单的程序设计一般包含以下几个部分。

（1）确定数据结构。

（2）确定算法。

（3）编码。

（4）在计算机上调试程序。

（5）整理并写出文档资料。

【答案】B

13.【解析】在本题中，熟练考试技巧的同学应该很容易发现答案在 A 和 B 之间，在计算机中，程序的运行过程都是以二进制方式进行的。因此本题答案选 B。

【答案】B

14.【解析】在选项 A 中，有取余运算符，该运算符的两操作数都必须为整数，根据题目对变量的定义，我们可以知道 a 是整数，而 x-y 应是实数，但这里用了一个强制转换，将 x-y 的结果转换为整型数据，因此该表达式是正确的表达式；对于选项 B，我们很容易发现它的错误，因为它后面用了语句结束标识分号，表明是一条语句，而非表达式；对于选项 C，由于 a*y 的结果是实数，所以该表达式不正确；对于选项 D，由于赋值运算符的左边不能是常量或表达式，因此错误。

【答案】A

15.【解析】本题考查我们对合法常量的鉴定，在选项 A 中，由于用了逗号，因此不能表示为整数；在选项 B 中给出的实数违背了实数指数表示时指数不能为小数的原则，因此错误；选项 C 中的‘\’不能单独表示字符，它一般和一些字母组合在一起表示一个字符，如‘\n’和‘\t’等；选项 D 的描述是正确的，“\007”是一个合法的字符串。

【答案】D

16.【解析】本题给出的表达式 a+=a-=a=9 其实等价于 a=a+(a=a-(a=9))，那么首先执行赋值操作将 a 的值变为 9，然后执行 a-a 操作，得到 0 赋值给 a，然后执行 0+0 操作，赋值给 a，因此 a 最终的结果为 0。

【答案】D

17.【解析】本题中题目给出的表达式 if(W)，W 的值可能取 0，表示假，也可能取其他值，表示真。选项 B 的表达式，由于用函数 getchar()，有可能取到任何字符即非 0，也有可能取到 0 即字符串结束标志，因此等价；选项 C 中完全可能取到 0 或非 0；而选项 D 根据变量 a 的值的变化也完全可能取到 0 或非 0。只有选项 A 是不行的，因为在 C 语言中没有运算符‘<>’。

【答案】A

18.【解析】在本题程序中，首先定义了变量 a 和 b 并进行了初始化，if 结构的执行次序是，首先执行 if(!a)，由于 a=1，因此执行第一个 else 后面的语句，即接着执行 if(a==0)，显然结果为假，根据 if 与 else 的配对原则，这时 if 结构语句结束，程序执行输出语句，那么变量 b 的值没有改变，因此最后输出的结果为 0。

【答案】A

19.【解析】本题主要考查 switch 结构。选项 A 中的条件 x%2 很明显是错误的，因为取余运算的操作数不能是实数；选项 B 中的条件(int)x/2.0 的结果是实数，但 switch 语句中的表达式和 case 后的表达式一般只能为整型、字符型或枚举型，因此也是错误的，这样我们还可以知道选项 D 也错误。因此本题答案选 C。

【答案】C

20.【解析】在程序中，首先定义了变量 a 和 b 并进行了初始化，接着执行 while 循环，程序进入循环。

第 1 次循环，首先将 b 变为 3，a 变为 3，由于 a 小于 6，执行第 2 次循环，这时将变量 b 变为 6，将变量 a 变为 5，这时仍然由于 a 小于 6，执行第 3 次循环，这时将变量 b 变为 11，而变量 a 变为 7，由于 11%10=1，所以最后 b 的值为 1。因此最后输出的结果为 7,1。

【答案】B

21.【解析】在本题的程序中，首先定义了变量 y 并初始化为 10；然后执行 while 循环，从题目给出的程序看，该循环体为空，因此当 y 的值等于 0 时，循环结束，不过这时将 y 的值再进行了一次自减操作，因此 y 的值为-1。所以最后输出的结果为 y=-1。

【答案】B

22.【解析】在本题的程序中，首先定义了一个字符数组，并进行了初始化，然后执行输出语句，输出的表达式为*s+2，而*s 取到的是数组中的第一个元素，即字符‘r’，然后加 2（这里计算的是其 ASCII 值），因此最后输出的是 t。

【答案】C

23.【解析】在本题程序中，首先定义了一个字符数

组并进行了初始化，然后将字符数组的首元素赋值为 0，将第 2 个元素赋值为‘\0’，将第 3 个元素赋值为字符‘0’，最后要求我们输出该字符数组所占的存储空间和字符数组中字符的长度。根据字符数组的定义我们知道，初始化时字符串有 6 个字符，占 6 个字节，然后加上一个系统默认的字符串结束标识，因此应该是 7 个字节，另外由于在后面将字符串的第一个字符赋值为 0，这正是字符串的结束标识，而用 strlen 求字符串的长度时碰到 0 结束，因此最后输出的结果是 7 0。

【答案】B

24.【解析】本题中考查的是函数的嵌套调用。在程序中首先声明了一个函数 f，从函数 f 的定义中，我们不难看出其作用是返回形参的 2 倍。在主函数中，三层嵌套调用函数 f，其参数值为 1，因此首次返回的是 2*1，然后返回的是 2*2*1，最后返回的是 2*2*2*1，所以最后输出的结果是 8。

【答案】D

25.【解析】本题中选项 A 中定义了一个指针变量 p，由于指针变量 p 本身就是地址值，而选项在前面加上一个取地址符，因此错误；选项 B 其本身没有语法错误，而且在一些编译系统（如 TC）是能通过编译并正确实现输入的，但在 VC 6.0 编译系统中是不允许这样定义的，因为指针变量没有指向一段已经分配的空间，因此无法完成输入操作；选项 C 是正确的定义形式；选项 D 的错误在于*p=&k 语句，因为*p 表示的是取值，而&k 表示取地址。

【答案】C

26.【解析】本题考查指针数组。在题目中首先定义了一个长度为 4 的整型指针数组。选项 A 定义的是一个普通的整型数组，而不是指针数组；选项 B 定义的是一个指向指针的指针，而并非指针数组；选项 D 定义的是一个指向数组的指针，而不是指针数组，由排除法我们可以知道本题答案选 C。事实上选项 C 定义的也是一个长度为 4 的指针数组，因为指针数组中的每个元素都是指针，而选项 C 中数组中各元素也是指针。

【答案】C

27.【解析】本题考查数组的定义。这里我们注意定义数组时，数组的长度应该是确定不变的，因此定义数组长度的表达式应该是一个常量或常量表达式。选项 A 使用了变量 N，显然不对，选项 C 的定义形式不正确，而选项 D 没有给出数组的长度，也不正确，因此正确的是选项 B。

【答案】B

28.【解析】在本题中，题目要求我们定义一个具有 5 个元素的整型数组。选项 A 定义数组时说明了其长度为 5，并都初始化为 0；选项 B 虽然没有说明其长度，但通过初始化操作确定了数组具有 5 个元素，并都初始化为 0，其意义和选项 A 一样；选项 C 定义的数组 c 通过常量表达式 2+3 说明数组的长度，这是正确的；选项 D 不正确，因为在定义数组长度时，使用的是变量 i。

【答案】D

29.【解析】在本题中，程序定义了一个函数 f，其形参为指针变量 p，函数体是将指针变量值加 3，然后再输出当前所指向元素的值。在主函数中，首先定义了一个一维数组，然后用指针变量 r 指向该数组，接着调用函数 f，实参为 r，根据函数 f 的功能，我们知道在函数 f 的输出语句中输出的是数组 a 的第 4 个元素，因此第一个输出的是 4，接着执行主函数中的输出语句，输出的是指针变量 r 所指向的元素值，由于 C 语言中形参并不能传值到实参，因此 r 仍然指向的是数组的首地址，因此输出的是 1。所以程序运行后输出的结果是 4,1。

【答案】D

30.【解析】在本题中，题目告诉我们函数 fun 只对下标为偶数的元素进行操作，从 f 的函数体中我们可以看出这是一个排序的过程，而且是将下标为偶数的元素按从大到小的顺序排列。在主函数中，首先定义了一个数组 aa 并进行了初始化，然后调用函数 fun，实参是数组名 aa 和整型变量 7，这我们就知道是对数组 aa 的前 7 个元素进行操作，那么根据 fun 函数的功能，排序后的 aa 应该为 7,2,5,4,3,6,1，所以本题应该选 A。

【答案】A

31.【解析】在本题中，选项 A 中的 strcmp 函数的功能是按照字典顺序判断两参数所指向字符串的大小，相等返回 0，否则返回非 0。如果 if 语句的条件为真，则执行 ST，那么说明字符串 s1 和字符串 s2 相等，这与题目意思一样，因此本题选 A。选项 B 中“s1= =s2”不能用来判断两字符串是否相等，选项 C 中的函数 strcpy 是用来复制字符串用的，而选项 D 中“s1-s2”是不对的，因为 C 语言中字符串是不能以一个整体进行做差运算的。

【答案】A

32.【解析】在本题中，选项 A 采用的是 while 循环结构，循环的条件是*t=*s，我们大家知道循环结束时判定条件为假，即 0，那么这里只有在赋值字符串结束标识‘\0’时，循环才会结束，而这个时候字符串的复制操作已经完成。选项 B 采用的是 for 循环，循环结束的条件其实和循环 A 一样，也是取到字符串结束标识符‘\0’时循环结束，因此也能完成字符串复制。选项 D 采用的是 for 循环，循环变量的初值都为 0，因此从第一个字符开始赋值，直到取到结束标识符‘\0’时，循环结束，这也能完成字符串的复制。因此可知选项 C 不正确。

【答案】C

33.【解析】在本题中，首先定义了一个数组 a 和 b 并进行了初始化，然后调用 strcat 函数，该函数的作用题目已经告诉我们，是连接两个字符串，程序最后输出连接

后的字符串。由于 strcat 函数读字符串时，碰到字符串结束标识符‘\0’结束，因此在读字符串 a 时，只读到第一个字符串结束标识符，然后将字符串 b 的内容连接到后面输出，因此最后的输出结果是 ABCDIJK。

【答案】B

34.【解析】在本题中，首先定义了一个 fun 函数，其形参为指针变量 p，用于指向待操作的字符串，从其循环体中我们可以看出，只要取到空格字符且其前一个字符是小写字母，那么就将该小写字母变成大写字母保存。在主函数中，首先定义了一个字符串数组 s1 并进行了初始化，然后调用 fun 函数，最后输出字符数组中的字符串。经过前面的分析，我们知道，被改变的字符必须满足两个条件，第一是本身必须是小写字母，第二是其后一个字符是空格字符。这样就很容易得出了最后输出的字符串应该是“aB cD EFG !”。

【答案】C

35.【解析】本题考查的是函数的递归调用，第一次调用函数 fun 时，函数的参数 x 是 7，由于 7/2=3，大于 1，因此递归调用函数 fun 此时的参数为 3，这时由于 3/2=1，不大于 1，因此这时输出参数 3 的值接着回到其上层 fun 函数执行，接着执行输出语句，这时输出的是参数 7，然后回到主函数执行，输出换行。因此整个程序输出的结果是 3 7。

【答案】D

36.【解析】在本题中，首先我们来分析一下 fun 函数，在 fun 函数中定义并初始化了一个静态变量 x，然后返回该静态变量 x 加 1 后的结果。在 C 语言中，对静态变量的赋值只在程序编译时进行，也就是说在整个程序运行期间只进行一次赋值。在主函数中，定义了一个变量 s 并初始化为 1，用户用一个循环来累加变量 s 的值，循环执行 5 次，第 1 次循环后，变量 s=1+2=3；第 2 次循环后，变量 s=3+3=6；第 3 次循环后，变量 s=6+4=10；第 4 次循环后，变量 s=10+5=15；第 5 次循环后，变量 s=15+6=21。因此最后输出的结果是 21。

【答案】B

37.【解析】在本题中，首先定义了 3 个同类型的指针变量，然后动态分配一个存储空间并使 3 个指针变量都指向这个存储空间，然后将 1 存放到该存储空间，然后将 2 存储到该存储空间，这时 1 已经被覆盖，同理当 3 存放到该空间时，2 又被覆盖，因此最后指针变量 a，b，c 都指向的空间元素值为 3，所以最后输出的结果是 3,3,3。

【答案】A

38.【解析】本题主要考查整型和双精度型在 VC 编译系统中所占的字节数。各类型所占的字节数是由编译系统决定的。整型在 VC 编译系统中占 4 个字节，在 TC 编译系统中一般占 2 个字节，而双精度型在 VC 编译系统中占 8 个字节，在 TC 编译系统中占的字节数也是 8。

【答案】C

39.【解析】在本题中，程序用到了关键字“typedef”，这个关键字的作用是声明新的类型名来代替已有的类型名。在本题中定义了 T 来代替结构名为 S 的结构体，因此在声明该结构体变量时，可以用 T 来进行，如“T a;”等价于“struct S a;”。因此本题的正确答案选 B。

【答案】B

40.【解析】本题主要考查位运算。本题要求输出的变量 c 的值为 248，其初值是 124，很显然这存在两倍的关系，而左移一位相当于乘以 2，因此在本题的下画线处应该填入的是左移 1 位，即“<<1”。

【答案】D

二、填空题

1.【解析】本题主要考查栈的出入操作，栈是一种先进后出的数据结构。题目告诉我们将 5，4，3，2，1 依次入栈，然后退栈一次，很显然这时退栈的是 1，接着将 A，B，C，D 入栈，然后全部退栈，其栈中各元素的退栈顺序是 DCBA2345。所以最后出栈的顺序应该是 1DCBA2345。

【答案】1DCBA2345

2.【解析】本题我们分两种情况说明：一种是无序的线性表，在这种情况下，要找 n 个数据中值最大的数据应该要和其他所有元素进行一次比较才能确定其值是最大的，如果有一个元素没比较，那么也不能确定当前元素是值最大的元素，因此至少需要比较的次数是 n-1 次；另一种是有序的线性表，在这种情况下，不管是升序还是降序线性表，其最大值的位置都是确定的，无须比较。当然本题考查的应该是第一种情况，因此答案为 n-1。

【答案】n-1

3.【解析】计算一棵二叉树结点个数的公式为：度为2的结点数*2+度为1的结点数*1+1。根据这个公式我们可以知道题目中描述的二叉树结点数为25个。

【答案】25

4.【解析】本题主要考查结构化程序的基本概念。仅由顺序、选择（分支）和重复（循环）结构构成的程序是结构化程序。

【答案】结构化

5.【解析】数据库的设计包括需求分析、概念设计、逻辑设计和物理设计 4 个阶段。

【答案】物理设计

6.【解析】本题主要考查不同进制数的输出。在本题的程序中，首先定义了整型变量 a 和 b，将 a 初始化为 200，将 b 初始化为八进制数 10，然后要求以十进制形式输出这两个数，八进制数 10 转换为十进制后为 8，因此最后输出的结果是 2008。

【答案】2008

7.【解析】本题主要考查我们对输入输出语句的掌握情况。程序中首先定义了两个整型变量，然后分别给其输入值，给 x 输入的输入格式是“%2d”，这表示取输入的两位，给变量 y 输入的输入格式是“%1d”，表示取输入的 1 位。最后程序输出的是变量 x 和 y 的和。从程序运行时的输入 1234567 我们可以知道，x 只取前两位，那么其值为 12，而 y 只取 1 位，其值为 3，因此最后的输出结果为 15。

【答案】15

8.【解析】本题主要考查逻辑值“真”和“假”。在 C 语言中，表达式的值为 0 时表示的是假，表达式的值为非 0 的时候，表示的是真。这里提醒大家不要简单地理解为 1。

【答案】非 0

9.【解析】在本题中，程序首先定于了一个数组 n 并初始化为 0，然后执行循环结构。

第 1 次循环时，输出 n[1]=n[0]*3+1=1；那么第 2 次循环时，输出 n[2]=n[1]*3+1=4；第 3 次循环时，输出 n[3]=n[2]*3+1=13；第 4 次循环时，输出 n[4]=n[3]*3+1=40。再根据输出的格式，每次输出的数字之间有一个空格，因此最后输出的是 1 4 13 40 。

【答案】14 13 40

10.【解析】本题题目告诉我们函数 fun 的功能是找出 N 个元素的一维数组中的最小值。从程序的返回语句可以看出，最小值存放在 x[k]中。空 10 所在的语句是 if 条件语句为真的情况下执行的语句，而 if 条件语句为真，说明所取到的当前数组元素值是小于 x[k]中的数值的，这个时候应该要将更小的这个值存放到 x[k]中，而这个更小值的数组下标为 i，因此第 10 空应该填“i”。

【答案】i

11.【解析】在本题程序中，定义了一个返回指针类型值的函数 f，该函数带有两个指针变量的形参，该函数的作用是返回两形参中值较大元素的地址。在主函数中，首先定义了整型变量 m 和 n，并分别初始化为 1 和 2，然后定义一个指针变量 r，使 r 指向变量 m，接着调用函数 f，参数分别为 r 和变量 n 的地址，并将函数 f 的返回结果赋值给 r。由于两实参分别是变量 m 和变量 n 的地址，而明显 n 的值要大于 m，因此函数 f 返回的是变量 n 的地址，所以在主函数中执行输出语句后输出的结果是 2。

【答案】2

12.【解析】本题题目要求我们找出二维数组中最大的元素作为函数值返回。在程序中，使用了一个二重循环来实现在二维数组中找最大值元素，从 if 结构中，我们不难看出 a[row][col]中存放着当前最大值元素，而循环结束后，a[row][col]中存放的应该就是整个二维数组中最大的元素值，所以第 12 空应该填写的就是 a[row][col]。

【答案】a[row][col]

13.【解析】在本题中，程序先用一个 for 循环结构对数组 n 进行赋值，且各元素都被赋值为 0，然后用一个双重循环进行运算。

当循环变量 i=0 时，循环变量 j=0 时，执行 n[0]=n[0]+1=1，循环变量 j=1 时，执行 n[1]=n[0]+1=1+1=2；

当循环变量 i=1 时，循环变量 j=0 时，执行 n[0]=n[1]+1=2+1=3，循环变量 j=1 时，执行 n[1]=n[1]+1=2+1=3；所以最后 n[0]与 n[1]的值应该都为 3。因此程序运行后的输出结果是 3。

【答案】3

14.【解析】在本题中，题目要求找出数组中的最大值并输出，那么从程序来看，只有第 14 空所在的这一条输出语句，那么第 14 空需要填写的就应该是最大值输出表达式，从前面的 if 结构不难看出，其当前最大值是用指针变量 s 指向的，那么在循环结束后，s 指向的值就是整个数组的最大元素。因此第 14 空应该填写*s。

【答案】*s

15.【解析】本题主要考查文件指针的定义。从后面的程序中我们可以看出 fp 是一个文件指针，那么第 15 空所在的语句就是用来定义文件指针的，因此我们可知第 15 空应该填写的是 FILE。

【答案】FILE